НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації та управління

УДК 004.422.81

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

Дипломний проект
на здобуття ступеня бакалавра

з напряму підготовки *6.050101 «Комп’ютерні науки»*

на тему: «*Комплекс задач підтримки логістичної діяльності малих підприємств*»

Виконав: студент *4* курсу, групи *ІС-52*

Голубець Богдан Віталійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *старший викладач Ковтунець О.В.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації *старший викладач Халус О.А.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент *доц. каф. ТК, к.т.н., доц. Ткач М.М.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації та управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) *6.050101*

«Комп’ютерні науки» («Інформаційні управляючі системи та технології»)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)

“___”_____2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Голубцю Богдану Віталійовичу
(прізвище, ім’я, по батькові)

1. Тема проекту «*Комплекс задач підтримки логістичної діяльності* малих підприємств»

керівник проекту *Ковтунець Олесь Володимирович, ст.викл.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “*23*”*квітня* 2019 р. №1181-с

2. Термін подання студентом проекту “*03*”*червня* 2019 року

3. Вихідні дані до проекту

Технічне завдання

4. Зміст пояснювальної записки

1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі

2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних

3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв’язання

4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів

5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1. Схема структурна діяльності

2. Схема структурна варіантів використань

3. Схема бази даних

4. Схема структурна класів програмного забезпечення

5. Схема структурна послідовності

6. Схема структурна компонентів програмного забезпечення

7. Рішення з математичного забезпечення

8. Креслення вигляду екранних форм

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *«15» лютого 2019 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	*Вивчення рекомендованої літератури*	*24.02.2019*	
2.	*Аналіз існуючих методів розв'язання задачі*	*27.03.2019*	
3.	*Постановка та формалізація задачі*	*30.03.2019*	
4.	*Розробка інформаційного забезпечення*	*02.04.2019*	
5.	*Алгоритмізація задачі*	*07.04.2019*	
6.	*Обґрунтування використовуваних технічних засобів*	*10.05.2019*	
7.	*Розробка програмного забезпечення*	*12.05.2019*	
8.	*Налагодження програми*	*15.05.2019*	
9.	*Виконання графічних документів*	*17.05.2019*	
10.	*Оформлення пояснювальної записки*	*18.05.2019*	
11.	*Подання ДП на попередній захист*	*30.05.2019*	
12.	*Подання ДП на основний захист*	*03.06.2019*	
13.	*Подання ДП рецензенту*	*05.06.2019*	

Студент _____ Б.В. Голубець
(підпис)

Керівник проекту _____ О.В. Ковтунець
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
			Документи текстові		
1	А4	ДП ІС-5206.1181-с.ПЗ	Пояснювальна записка	64	
2	А4	ДП ІС-5206.1181-с.ТЗ	Технічне завдання	9	
3	А4	ДП ІС-5206.1181-с.ПМВ	Програма та методика випробувань	13	
			Матеріали графічні		
4	А4	ДП ІС-5206.1181-с.ССД	Схема структурна діяльності	1	
5	А4	ДП ІС-5206.1181-с.ССД	Схема структурна варіантів використань	1	
6	А4	ДП ІС-5206.1181-с.СБД	Схема бази даних	1	
7	А3	ДП ІС-5206.1181-с.ССК	Схема структурна класів програмного	1	
			забезпечення		
8	А4	ДП ІС-5206.1181-с.ССП	Схема структурна послідовності	1	
9	А4	ДП ІС-5206.1181-с.ССК	Схема структурна компонентів	1	
			програмного забезпечення		
10	А3		Рішення з математичного забезпечення	1	
11	А3	ДП ІС-5206.1181-с.КЕ	Креслення вигляду екранних форм	1	

Зм.	Арк.	ПІП	Підп.	Дата	ДП ІС-5206.1181-с.ВДП		
Розробн.		Голубець Б.В.			Відомість дипломного проекту	Літ.	Лист 1 / Листів 1
Керівн.		Ковтунець О.В.					
Консульт.						КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52	
Н/контр.		Халус О.А.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проекту

на тему: Комплекс задач підтримки логістичної діяльності малих підприємств

Київ – 2019 року

АНТОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з п’яти розділів, містить 20 рисунків, 23 таблиці, 1 додаток, 24 джерела.

Дипломний проект присвячений розробці комплексу задач підтримки логістичної діяльності малих підприємств, тобто написанні мобільного застосування, який зможе не тільки вирішувати проблеми з плануванням та контролем дотримання маршруту, але ще й буде безкоштовним.

Для розгляду загальних положень було описано процеси діяльності, варіанти використання, описано та проаналізовано існуючі аналоги, та коректно поставлено задача із визначеними метою та задачами.

Для опису інформаційного забезпечення було визначено вхідні та вихідні дані, було створено та описано базу даних.

Для математичного забезпечення було знайдено дуже простий, та швидкий алгоритм пошуку найкоротшого шляху.

Для визначення програмного забезпечення було описано засоби розробки, висунуто вимоги до технічного забезпечення, обрано архітектуру програмного забезпечення.

Була описана інструкція користувача та проведено тестування комплексу задач.

МАЛІ ПІДПРИЄМСТВА, ЛОГІСТИЧНА ДІЯЛЬНІСТЬ, ПЛАНУВАННЯ МАРШРУТІВ, КОНТРОЛЬ ПЕРЕВЕЗЕНЬ

ABSTRACT

Structure and scope of work. The explanatory note of the diploma project consists of five sections, containing 20 figures, 23 tables, 1 supplement, 24 sources.

The diploma project is devoted to the development of a complex of tasks supporting the logistical activity of small enterprises, that is, the writing of a mobile application that can not only solve the problems with scheduling and controlling the compliance of the route, but will also be free.

To review the general provisions, processes of activity, variants of use were described, existing analogues were described and analyzed, and a problem with the defined goals and tasks was set up co-rectly.

In order to describe the information support, input and output data were determined, a database was created and described.

For mathematical support, a very simple and fast search algorithm was found for the shortest path.

To determine the software, the ro-design tools have been described, technical requirements have been put forward, software architecture has been selected.

A user manual was described and a complex set of tasks was tested.

SMALL ENTERPRISES, LOGISTIC ACTIVITY, ROAD PLAN, TRANSPORT CONTROL

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

HTTP запит – запити передачі даних у комп’ютерних мережах за протоколом HTTP. Назва скорочена від Hyper Text Transfer Protocol, протокол передачі гіпер-текстових документів.

Json - це текстовий формат обміну даними між комп'ютерами. JSON базується на тексті, може бути прочитаним людиною. Формат дозволяє описувати об'єкти та інші структури даних. Цей формат головним чином використовується для передачі структурованої інформації через мережу.

TSP - Travelling Salesman Problem, що в перекладі означає – задача комівояжера.

Асимптота – пряма, до якої крива при віддаленні в нескінченність наближається як завгодно близько.

Гамільто́нів шля́х — шлях, що містить кожну вершину графа рівно один раз. Гамільтонів шлях, початкова і кінцева вершини якого збігаються, називається гамільтоновим циклом.

Криптографія - наука про математичні методи забезпечення конфіденційності, цілісності і автентичності інформації.

Крос-платформенність – здатність програмного забезбечення працювати з двома і більше апаратними платформами.

Мале підприємство - мале підприємство - це фірма, якою керує незалежний власник, яка не посідає у своїй галузі чільного місця та відповідає певним критеріям за кількістю зайнятих і щорічним критеріям за обсягом доходу.

Маркер – відмітка на карті.

Нативна розробка – розробка з використанням лише рідних для платформи засобів розробки, протилежність до крос-платформенної розробки.

Реляційна база даних – база даних, заснована на реляційній моделі даних. Для роботи з реляційними БД застосовують реляційні СКБД. Інакше кажучи, реляційна база даних – це база даних, яка сприймається користувачем як набір нормалізованих відношень різного ступеня.

Складність обчислювальних процесів – це поняття теорії складності обчислень, оцінка ресурсів (зазвичай часу) необхідних для виконання алгоритму.

Хмарні обчислення – модель забезпечення повсюдного та зручного доступу на вимогу через мережу до спільного пулу обчислювальних ресурсів, що підлягають налаштуванню (наприклад, до комунікаційних мереж, серверів, засобів збереження даних, прикладних програм та сервісів), і які можуть бути оперативно надані та звільнені з мінімальними управлінськими затратами та зверненнями до провайдера.

ВСТУП

За останні роки значно зросла кількість малих підприємств, які тим чи іншим чином пов’язані з логістикою. Шлях розвитку таких підприємств тяжкий та довгий. Це пов'язано з тим, що кількість штатних співробітників не перевищує 50 осіб. Тому, роботу таких підприємства потрібно якось оптимізувати та полегшити.

Для пришвидшення розвитку малих підприємств необхідно автоматизовувати якомога більше різних бізнес процесів. Але для автоматизації виробництва потрібні значні кошти, яких малі підприємства, валовий дохід яких за рік не перевищує 70 млн.грн., напросто не має.

Таким чином постає задача в автоматизації логістичної діяльності шляхом зменшення затрат на планування перевезень. В сучасному світі існує багато способів це зробити, але найпоширеніші з них комплексні системи є дуже дорогими, а безкоштовні застосування не завжди гарантують точність, зручність та безпеку.

Саме тому було вирішено розробити мобільне застосування, яке би виконувало функції планування та контролю дотримання маршруту, та було б зручним для використання як керівником підприємства, так і для його співробітників.

Основні можливості цього застосування мають полягати у побудові оптимальних маршрутів, які мають 3 і більше обов’язкових пунктів, а також надання можливості контролю перевізника за дотриманням побудованих керівником маршрутів.

Дипломний проект присвячений розробці комплексу задач підтримки логістичної діяльності малих підприємств.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

На сьогоднішній день багато малих підприємств мають потреби у перевезенні вантажу різного типу.

Такі підприємства як правило не купують комплексні системи, які будуть налаштовані спеціально під них, тому що це зазвичай дуже дорого, а використовують існуючі загальні застосування, в більшості випадках ті, які є безкоштовними. Але такі застосування не завжди зручні та не можуть задовольнити усіх потреб щодо планування маршрутів.

Основним функціональним рішенням цього додатку є те, що він дозволить таким підприємствам швидко планувати та будувати маршрут, який включає в себе проміжні пункти, буде супроводжувати співробітника, який виконує перевезення у режимі реального часу, та надаватиме можливість керівнику контролювати просування перевізника по маршруту

Даний проект націлений на допомогу таким підприємствам з плануванням маршрутів, та контролю перевезень з метою зменшення витрат націлених на логістичну діяльність підприємства.

1.1.1 Опис процесу діяльності

Об’єктом автоматизації є процес планування та оптимізації маршруту. Система призначена для планування та контролю дотримання перевізником маршруту.

Послідовність дій при роботі з застосуванням представлена у вигляді схеми структурної діяльності та наведена у графічному матеріалі.

Для того щоб користувачі могли користуватися програмою необхідно спочатку зареєструватися під певним типом співробітника «Керівник», тип співробітника «Перевізник» може бути зареєстрованим лише керівником.

Керівник має право будувати та переглядати маршрути, закріпляти маршрути за перевізниками, а також відслідковувати дотримання маршруту перевізниками.

Перевізник має право лише переглядати маршрут, який закріпив за ним керівник, та звірятися з ним, тобто використовувати застосування як навігатор.

1.1.2 Опис функціональної моделі

Акторами системи є: керівник та перевізник. Визначимо, які функції вони виконують в системі, для цього наведемо таблицю 1.1, в якій описані актори, функції та їх описи.

Таблиця 1.1 – Опис функцій системи

Актор	Функція	Опис функції
Усі актори	1. Авторизація	Актору надається форма авторизації на стартовій сторінці. Актор вводить логін та пароль
Керівник	2. Реєстрація	Керівник реєструється у системі під типом користувача «Керівник», для реєстрації у системі йому надається вікно реєстрації, де він повинен вказати логін, пароль, ПІБ, e–mail, номер телефону, та назву підприємства
	3. Ведення перевізників	Керівник реєструє у системі всіх своїх перевізників, для цього йому надається вікно реєстрації перевізника, де він повинен вказати логін, пароль та ПІБ перевізника. Також керівник може видаляти перевізників
	4. Побудова маршруту	Керівник може побудувати маршрут, для цього він повинен вказати всі необхідні пункти маршруту на мапі у послідовності від пункту старту до кінцевого пункту, після чого він отримає побудований маршрут на мапі

Продовження таблиці 1.1

Актор	Функція	Опис функції
Керівник	5. Закріплення маршруту за перевізником	Одразу ж після побудови маршруту керівник має можливість закріпити побудований маршрут за перевізником, обравши необхідного перевізника, після чого цей маршрут буде відображатися у системі для перевізника
	6. Контроль дотримання маршруту	Керівник має можливість відслідковувати дотримання перевізником маршруту, у разі відхилення перевізника від маршруту система повідомляє про це керівнику, також керівник може бачити весь маршрут, та позицію перевізника на ньому
Перевізник	7. Перегляд маршруту	Перевізник може лише переглядати побудований та закріплений за ним керівником маршрут, та використовувати дане застосування як навігатор

Відповідно до визначених функцій у графічному матеріалі представлено схему структурну варіантів використань.

1.2 Огляд наявних аналогів

На сьогоднішній день існую багато програм з планування маршруту, перелічимо основні з них, їх переваги та недоліки.

«Google maps» – безкоштовний картографічний сервіс від компанії Google, а також набір додатків, побудованих на основі цього сервісу й інших технологій Google. На сьогоднішній день можна сказати що це мабуть найбільш популярний сервіс з планування маршруту, цей сервіс є кросплатформеним, має зручний графічний інтерфейс, має в собі довідник громадського транспорту, на основі якого може будувати маршрути, аналізує затори на

дорозі, через нього можна навіть подивитися змодельоване за допомогою фотографій зовнішнє середовище за відповідними координатами на мапі, але він має основний недолік у тому що він не може побудувати оптимальний маршрут по всім потрібним пунктам. Графічний інтерфейс Google maps представлений на рисунку 1.1.

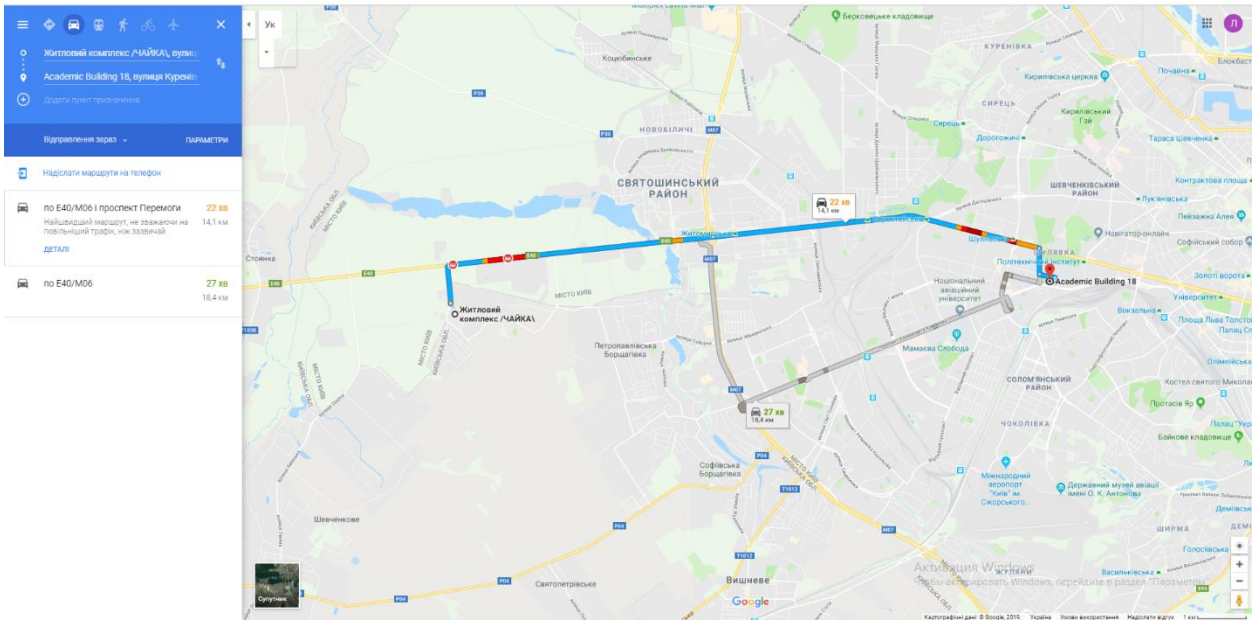


Рисунок 1.1 – Графічний інтерфейс Google maps(веб версія)

«Яндекс. Навігатор» — безкоштовна навігаційна мобільна програма від компанії Яндекс. Доступна для смартфонів та планшетів на платформах iOS, Android та Windows Phone. У порівнянні із Google maps має значно менше функціоналу, ця програма може лише планувати маршрут та аналізувати затори, але на мою суб'єктивну думку Яндекс. Навігатор прокладає значно якісніші для користувача маршрути ніж Google maps (критерії якості: довжина маршруту, та якість дорожнього покриття), раніше навіть мав сервіс, який дозволяв прокладати оптимальні маршрути через декілька проміжних пунктів, але цей сервіс був незручний для користувача. Основним недоліком Яндекс. Навігатора є його недоступність у нашій країні, через політичні обставини, щоб його використовувати– треба використовувати VPN, що

заборонено законом, та дуже сповільнює передачу даних на пристрої.

Графічний інтерфейс Яндекс Навгітору зображений на рисунку 1.2.

Рисунок 1.2 – Графічний інтерфейс Яндекс. Навігатор

«2GIS» — це оффлайн карти та довідник з контактами фірм міста. Додаток працює без підключення до мережі інтернет: адреси, телефони, час роботи компаній доступні в будь–який момент. Пошук проїзду на громадському транспорті або на особистому автомобілі допоможе дістатися до знайденого по найбільш якісним маршрутам. Має зручний графічний інтерфейс, має багато варіантів маршруту з точки А у точку В, здатний знайти маршрут навіть там, де нема дороги. Недоліком є те, що ним користується мало людей, а отже інформація щодо заторів на дорогах зазвичай хибна. Графічний інтерфейс 2GIS представлений на рисунку 1.3.

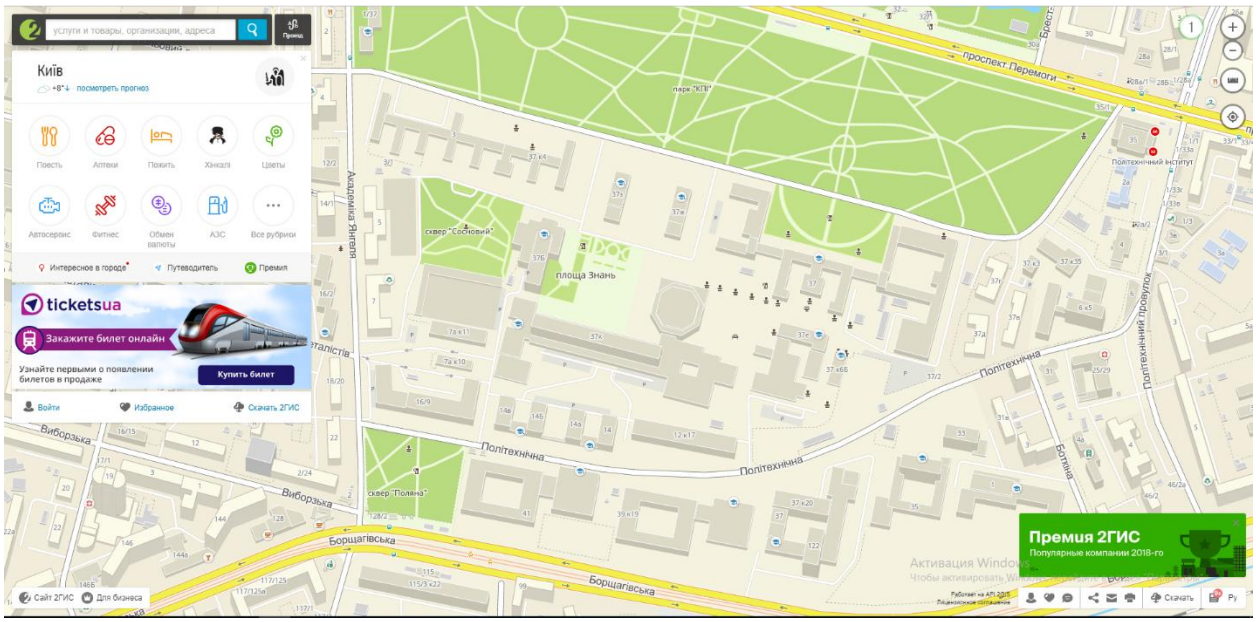


Рисунок 1.3 – Графічний інтерфейс 2GIS (веб версія)

Російський сервіс **«Логіст»** – логістичний сервіс планування маршрутів, який здатен будувати маршрути з багатьма проміжними пунктами, але він не може ніяк допомогти користувачу у режимі реального часу, та не може надати керівнику можливості слідкувати за перевізниками, не має мобільної версії, а також цей сервіс розрахований на планування перевезень між містами, використовувати його для перевезень в рамках одного міста недоречно. Графічний інтерфейс сервісу «Логіст» представлено на рисунку 1.4 [5].

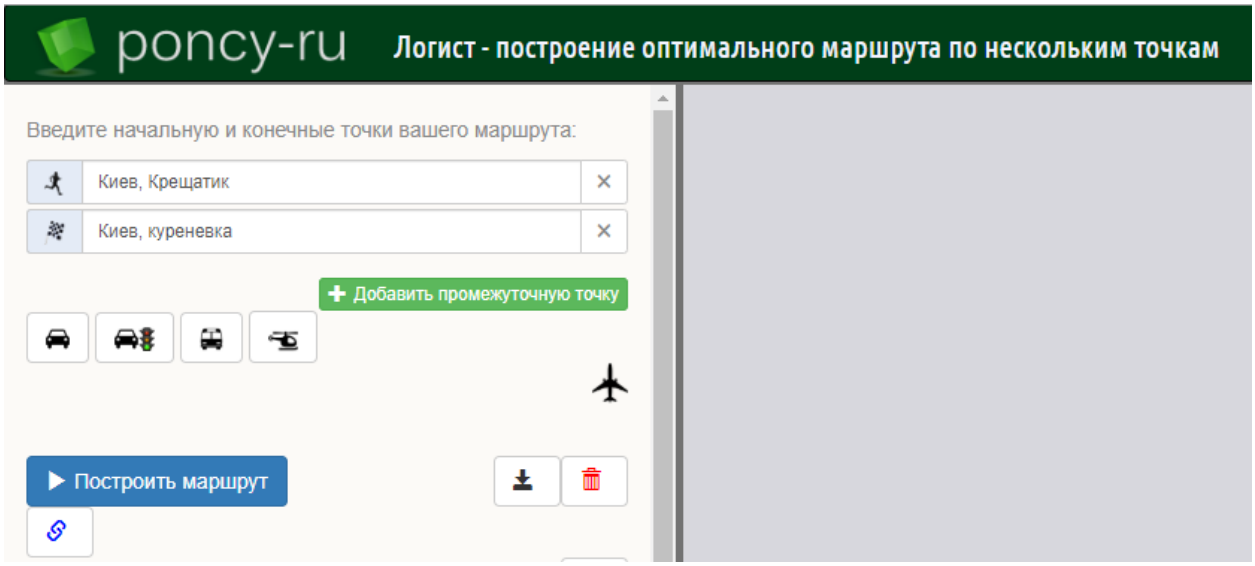


Рисунок 1.4 – Графічний інтерфейс сервісу «Логіст»

«Speedy Route» – картографічний сервіс побудови маршрутів, найбільш поширений за кордоном. В описі продукту сказано, що сервіс розраховує найбільш ефективний по кількості витрат палива маршрут між декількома локаціями, де вихідна і кінцева точка єдині. Цей сервіс має дуже багато недоліків, назвемо декілька з них:

- при введенні даних виникають труднощі перекладу назв вулиць та міст на англійську мову;
- у сервісі передбачено мінімальну кількість пунктів маршруту – 5; маршрути, що складаються з 4 пунктів побудувати неможливо;
- на сайті представлена урізана версія мапи для ознайомлення, для доступу до повної версії потрібно оформити платну підписку;
- сервіс не має мобільної версії, та не може бути корисним у режимі реального часу.

Графічний інтерфейс сервісу «Speedy Route» представлений на рисунку 1.5 [6].

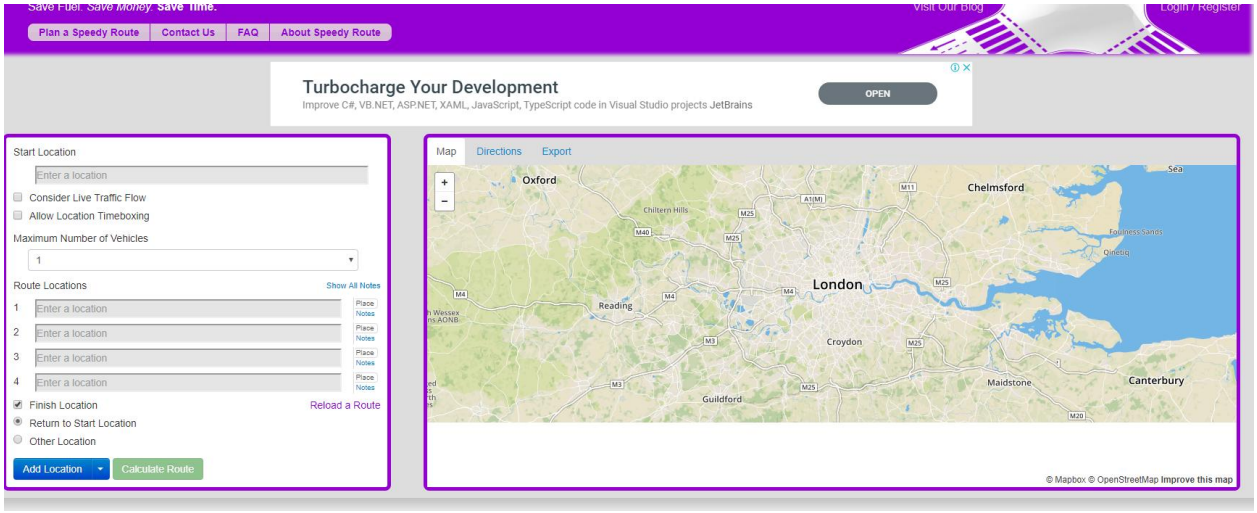


Рисунок 1.5 – Графічний інтерфейс сервісу «Speedy Route»

1.3 Постановка задачі

1.3.1 Призначення розробки

Призначенням системи є забезпечення простого та швидкого планування маршрутів перевезень для малих підприємств та можливість легко відстежувати ці перевезення.

1.3.2 Цілі та задачі розробки

Цілями розробки є:

– спростити планування перевезень для малих підприємств;

– надати малим підприємствам можливість контролювати всі свої перевезення в режимі реального часу.

Для досягнення поставлених цілей необхідно розв’язати наступні задачі:

– авторизація користувачів у системі;

– швидка побудова оптимального маршруту та відображення його на мапі;

– відстеження перевізника в режимі реального часу.

Висновок до розділу

У даному розділі здійснений детальний аналіз предметної області, визначені актори системи. Оглянуто та проаналізовано існуючі аналоги. Визначено цілі задачі розробки.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Вхідні дані представлені у таблиці 2.1.

Таблиця 2.1 – вхідні дані

Данні	*Опис*
Дані реєстрації	Дані користувачів, які заповнюються у формі реєстрації(Ім'я, Прізвище, По–батькові, логін, електронна адреса, пароль, телефон, назва підприємства). Слугують для реєстрації користувача у системі
Дані авторизації	Дані зареєстрованого користувача, які заповнюються у формі авторизації користувача при вході в систему(Логін, пароль). Слугують для входу в систему користувача
Пункти для побудови маршруту	Дані, які заповнює користувач, обираючи пункти на мапі. Слугують для побудови оптимального маршруту. Для користувача подаються у вигляді міток на мапі

2.2 Вихідні дані

Вихідні дані представлені в таблиці 2.2.

Таблиця 2.2 – вихідні дані

Дані	*Опис*
Оптимальний маршрут	Дані, які отримує користувач після введення вхідних даних, та пошуку по ним маршруту, представлені у вигляді прокладеного маршруту на мапі

2.3 Опис структури бази даних

Оскільки ми використовуємо базу даних з хмари – вона не реляційна, дані в цій базі зберігаються як JSON. Дані JSON записуються парою ім’я/значення. Для зручної роботи з даними виділимо типи даних, які нам потрібні. Така візуалізація даних представлено в таблиці 2.3.

Таблиця 2.3 – Опис даних, які зберігатимуться у хмарі

Ім’я	Опис типу значення, яке зберігається
Users	Колекція користувачів типу керівник
Manager	Має такі поля: CompanyName, Email, Id, Login, Surname, Name, Patronymic, Password, Phone, Driwers, значення яких буде описано далі у таблиці
Companyname	Назва фірми (текст)
Email	Електронна адреса (текст)
Id	Числовий ідентифікатор користувача (ціле число)
Name	Ім’я користувача (текст)
Login	Логін користувача (текст)
Surname	Прізвище користувача (текст)
Patronymic	По – батькові користувача (текст)
Password	Пароль користувача (текст)

Продовження таблиці 2.3

Ім’я	Опис типу значення, яке зберігається
Phone	Номер телефону користувача (текст)
Drivers	Колекція користувачів типу перевізник
Driver	Користувач типу перевізник, має такі поля: CompanyName, Email, Id, Login, Surname, Name, Patronymic, Password, Phone, значення яких вже було описано вище, та поля: Route, CurrentPossition, значення яких буде описано нижче
Route	Зашифрований у рядок маршрут (текст)
CurrentPossition	Координати поточного місця знаходження перевізника, складається з двох полей X, та Y (числа)

Отже для заповнення JSON у хмарі будемо використовувати приведені вище типи даних. Тепер побудуємо схему структури зберігання даних у хмарі. Схема структури Схема бази даних наведена у графічному матеріалі.

Висновок до розділу

У даному розділі були описані вхідні/вихідні дані, був описаний тип сховища хмари, у якому зберігаються дані. Було придумано, та описано які типи даних будуть використовуватись для збереження даних, також було спроектовано схему структури зберігання даних у хмарі.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Застосування повинне допомагати малим підприємствам вирішувати їх логістичні проблеми, а саме проблеми пов'язані з плануванням маршруту. У даному випадку існує 2 типи користувачів, перший тип користувача – це керівник, тобто співробітник підприємства, відповідальний за всі перевезення у підприємстві, другий тип користувача – це перевізник, його задача полягає лише у перевезенні товару.

Задача керівника полягає у плануванні оптимального маршруту по цільовим пунктам, та передачі цього маршруту перевізнику. Також керівник повинен мати можливість відслідковувати поточне місце знаходження перевізника на маршруті. Місце знаходження перевізника може бачити лише його керівник.

Застосування повинне супроводжувати перевізника у дорозі, тобто слугувати для нього навігатором.

3.2 Математична постановка задачі

Математична постановка задачі має наступний вигляд:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\, x_{ij} \rightarrow \min \ (3.1) \text{ при обмеженнях:}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \ (i = \overline{1,n}) \ (3.2) - \text{обмеження на одноразовий виїзд з пункта;}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \ (j = \overline{1,n}) \ (3.3) - \text{обмеження на одноразовий в`їзд до пункта;}$$

Де c_{ij} - матриця відстаней між усіма пунктами $i, j = \overline{1, n}$.

Якщо в моделі задачі обмежитися лише умовами (3.2) і (3.3), то вона буде еквівалентною задачі про призначення, план якої не обов'язково повинен бути циклічним. Тобто, маршрут перевізника може розпастися на декілька незв'язних між собою циклів, тоді як насправді він повинен складатися з одного циклу. Щоб забезпечити цю вимогу введемо наступне обмеження:

$$u_i - u_j + nx_{i,j} \leq n - 1;\ i, j = \overline{1, n};\ i \neq j \quad (3.4)$$

Покажемо, що в довільному циклі, який починається в першому пункті, можна знайти такі u_i та u_j , які задовольняють нерівність (3.4). Нехай на k-му кроці перевізник переміщується з пункта i в пункт j. І припустимо, що $u_i = k$. Далі, на $k + 1$-му кроці перевізник буде вирушати з j-го пункта в наступному напрямку, тоді $u_j = k + 1$. Якщо підставити дані величини в (3.4), отримаємо:

$$u_i - u_j + nx_{i,j} = k - (k + 1) + nx_{i,j} = -1 + nx_{i,j} \leq n - 1$$

Зауважимо, що дана нерівність виконується для будь-яких значень i та j при $x_{ij} = 0$. Якщо ж $x_{ij} = 1$, то нерівність (3.4) виконується як строга рівність:

$$u_i - u_j + nx_{i,j} = k - (k + 1) + n = n - 1$$

Тобто, якщо перевізник пересувається з i-го в j-тий пункт, то нерівність (3.4) фіксує порядкові номери цих міст.

Отже, математична постановка задачі полягає у мінімізації функції (3.1) при обмеженнях (3.2), (3.3) і (3.4). Якщо немає жодних обмежень, то постановка задачі полягає лише у мінімізації функції (3.1).

3.3 Обґрунтування методу розв’язання

В цьому розділі розміститься аналіз існуючих алгоритмів TSP з описом їх переваг та недоліків.

3.3.1 Аналіз алгоритмів пошуку оптимального шляху

Алгоритми для розв’язку поставленої задачі можна розділити на точні (exact Algorithm) та неточні (none-exact Algorithm). Точні алгоритми включають в себе перебір всіх можливих варіантів, в окремих випадках розв’язки можуть бути швидко знайдені, але в цілому здійснюється перебір n! циклів. Другі в загальних випадках застосовуються для задач, які неможливо вирішити точно (обчислення певних інтегралів, розв’язок нелінійних рівнянь, пошук квадратного кореня ...), якщо існуючі точні розв’язки вимагають значних і невиправданих витрат часу при високій складності задачі, і як частина більш складного алгоритму, за допомогою якого задача вирішується точно [7].

3.3.1.1 Точні алгоритми

У свою чергу існує дві групи точних алгоритмів - одна з них використовує методи релаксації лінійного програмування TSP: алгоритм Гоморі, метод внутрішньої точки, метод гілок та меж; друга, менша група, використовує методи динамічного програмування. Характерна особливість методів обох груп - гарантія знаходження оптимальних розв’язків при загальній трудомісткості процесу [8].

Повний перебір (Brute Force). Один із найочевидніших методів пошуку оптимального шляху – метод повного перебору, або метод грубої сили. Суть цього методу полягає у переборі усіх можливих варіантів, алгоритм можна записати таким чином:

- знайти загальне число можливих Гамільтонових контурів;

– знайти вагу кожного Гамільтонового контуру, склавши вагу усіх його ребер;
– обрати Гамільтонів контур з мінімальною вагою, який буде оптимальним.

Метод повного перебору має низку переваг – він гарантує знаходження розв’язку задачі TSP, при цьому він прямолінійний і простий у виконанні. У той же час, алгоритм вважається неефективним при роботі з великим обсягом даних, так як для знаходження оптимального маршруту вимагає знайти вагу (n - 1)! Гамільтонових контурів.

На таблиці 3.1 зображена кількість часу, необхідного для розв’язку задачі методом повного перебору при потужності обчислювальної машини, яка дозволяє обчислювати 1 мільйон Гамільтонових контурів в секунду.

Таблиця 3.1 – Розрахунковий час розв’язку TSP методом повного перебору [9]

Кількість пунктів	Обчислювальний час
10	1/3 секунди
13	8 хвилин
15	1 рік
20	193 роки

Метод гілок та меж. Метод гілок та меж часто використовується для знаходження оптимального розв'язку задач комбінаторної оптимізації. Його суть полягає в розбитті множини на підзадачі і виключенні свідомо неоптимальних розв'язків.

Нехай граф V містить усі цілові пункти, П – множина усіх перестановок пунктів, покриваюча усі можливі розв’язки. Розглянемо перестановку

$\pi \in \Pi$, в якій кожному пункту ставиться у відповідність наслідник – і для πі пункта. Таким чином обхід можна записати як (1, π(1), π (π (1)), … , 1). Якщо число пунктів в обході дорівнює n, тоді перестановку називають циклічною. Задача про призначення ставить перед собою ціль знайти циклічні перестановки, а наша задача переслідує ту же ціль, але з обмеженням, що у цих перестановок має бути мінімальна вартість. Метод гілок та меж в першу чергу знаходить розв'язок задачі про призначення, складність якої для n пунктів доволі велика і асимптотично рівна $O(n^3)$ [10].

Якщо був знайдений повний обхід, то отримане значення також є розв'язком задачі. В іншому випадку проблема поділяється на декілька підгалузей, кожна з яких виключає деякі дуги частини обходу, таким чином виключаючи саму частину обходу. Метод, за допомогою якого вираховується, яку дугу слід видалити, називають правилом розгалуження. Важливе зауваження - не повинно існувати дубльованих підзадач, їх загальна кількість має бути мінімізована.

Будемо використовувати критерій, гарантуючий незалежність підзадач – розглядається включений набір дуг, які не належать набору. Позначимо *E* як множину виключених дуг та *I*, як множину включених. Розкладемо *I*. Оберемо *t* дуг підгалузей $x_1x_2 \dots x_n$ які не належать *I*. Задача розділена на *t* підзадач так, щоб у j_{th} підзадачі були E_j виключених дуг та I_j включених дуг. Запишемо у вигляді формули:

$$k = \overline{1,J} \begin{cases} E_j = E \cup \{x_j\} \\ I_j - I \cup \{x_1, x_2, \dots x_{j-1}\} \end{cases}$$

Але x_j – виключена дуга j_{th} підзадач та включена дуга у $(j + 1)_{st}$ області. Це означає, що обхід отриманий розв'язком $(j + 1)_{st,}$ не містить цю дугу. Це гарантує відсутність однакових маршрутів.

Кількість можливих розв’язків дорівнює $(n - 1)!/2$ для $n = 50$ це приблизно 3*1062. Цей метод найбільш часто використовується при кількості вузлів від 40 до 60 [11].

Необхідність цілком вирішувати завдання лінійного програмування у всій області допустимих розв’язків можемо вважати головним недоліком вищеописаного методу. Для задач з великим об'ємом даних метод гілок та меж є невиправдано трудомістким, в той же час алгоритм є надійним методом вирішення цілочислових задач.

Алгоритм Гоморі (The Cutting Plane). У 1954 році була представлена робота Данцига, Фалкерсона і Джонсон, що описує новий метод розв'язання задачі комівояжера, який також може бути використаний для вирішення будь-якої проблеми.

$$minimize\ c^T x\ subject\ to\ x \in S$$

де $c \neq 0$, S – кінцева підмножина деякого R^m, і таким чином ми зможемо знайти точки S. Це ітераційний алгоритм – кожний повтор починається з лінійної програмної релаксації. Запишемо у вигляді формули:

$$minimize\ c^T x\ subject\ to\ Ax \leq b$$

де многогранник P, визначений як $\{x : Ax \leq b\}$, містить S та обмежений. Так як P обмежений, ми зможемо знайти оптимальний розв’язок x^* як екстремальну точку P. Якщо x^* належить S, то оптимальний розв'язок знайдено, в противному випадку деяка лінійна нерівність задовільняє усі точки S та порушує x^*. Таку нерівність називають алгоритмом Гоморі, який детально описав його 1958, методом відсікаючих площин чи просто відсічень [12].

Даний метод використовується для побудови точних або наближених задач, особливо часто зустрічається в поєднанні з методом гілок та меж і тоді називається методом гілок та відсікань. Обидва методи засновані на

вирішенні послідовності релаксованих підзадач лінійного програмування. В алгоритмі Гоморі релаксовані підзадачі поступово покращують апроксимацію цілочисельної задачі, зменшуючи окіл оптимального розв'язку. Якщо оптимальність не вдалося отримати, тоді шукається наближений розв'язок з похибкою.

У методу відсікань є перевага над методом гілок та меж - перші більш зручні для апаратного обчислення, так як для їх розв'язку не потрібен великий обсяг оперативної пам'яті для зберігання дерева рішень [13].

Метод динамічного програмування (Dynamic Programming). Розглянемо задачу з n пунктами та відстанями d_{ij} між будь-якими двома пунктами, шлях починається та закінчується в пункті n_0. TSP може бути розв'язана за час O(n!) методом повного перебору, але алгоритм динамічного програмування дозволяє скоротити час до $O(n^2 2^n)$.

Позначимо підзадачу: нехай S буде підмножиною пунктів, які містять 1 і як мінімум ще один пункт, j буде пунктом, який відрізняється від 1, позначимо через C(S,j) найкоротший шлях, який починається в 1, та проходить через усі пункти S та закінчується в j. Запишемо алгоритм у вигляді псевдокоду.

$$\textit{for all } j \textit{ do } C(\{1,j\}, j) = d_{1j}$$

for s = 3 to n do (the size of the subsets considered this round)

for all subsets S of {1, ...n} of size n and containing 1 do

$$\textit{for all } j \in S, j \neq 1 \textit{ do}$$

$$C(S;j) = \min_{i \neq j \in S} [C(S - \{j\}, i) + d_{ij})]$$

$$Opt = \min_{j \neq 1} [C(\{1,2,\ldots n\}) + d_{j1}]$$ [14]

Даний метод використовується для підвищення ефективності обчислювальних повторень, зберігаючи проміжні результати і знову використовуючи їх при необхідності.

3.3.1.2 Неточні алгоритми

В цілому алгоритми даної групи пропонують потенційно неоптимальні, але швидкі розв'язки. У свою чергу наближені алгоритми можна розділити на дві категорії:

- наближені (Approximation Algorithms);
- евристичні (Heuristic Algorithms).

Алгоритм Кристофідеса (Christofides' Algorithm). Алогорітм Крістофідеса використовується для вирішення метричних TSP - з додатковою умовою, що для матриці відстаней виконано нерівність трикутника:

$$\forall\, i, j, k\ d_{ik} \geq d_{ij} + d_{ik}$$

Велика частина евристичних алгоритмів належать до наближеному класу. Професор Нікос Крістофайдс в 1976 році допрацював один з існуючих алгоритмів (метод подвійного мінімального остовного дерева, $O(n^2\log_2(n))$) так, що час розв'язку задачі не перевищує оптимальний час більш ніж на 3/2 [15].

Розв'язок оригінального алгоритму можна записати так: знайти мінімальне дерево з безлічі всіх пунктів, продублювати всі ребра і побудувати граф Ейлера, побудувати гамільтонів цикл, пройшовши кожен вузол тільки один раз і обрати найліпший шлях, що веде з кожного вузла.

Алгоритм Крістофідеса складається із послідовності наступних дій:

- знайти мінімальне дерево із множини всіх пунктів;
- знайти пари з мінімальною вагою множини вершин непарного ступеня і побудувати ейлерів граф;

- знайти ейлерів обхід і побудувати гамільтонів цикл, уникаючи вже відвіданих вузлів [15].

Основна відмінність - додаткове обчислення пари з мінімальною вагою. Ця частина також найбільш трудомістка, тому час виконання алгоритму зростає до $O(n^3)$. Проведені тести показали, що алгоритм Крістофідеса на 10% вище нижньої межі Хелд-Карп [16].

Алгоритм найближчого сусіда (NearestNeighbour). Один з найпростіших евристичних методів рішення TSP. Головне правило алгоритму - завжди вибирати сусіднє місто (сусіда). Рішення завдання складається з наступних кроків:

- обрати довільний пункт;
- знайти найближчий пункт, не включений у маршрут та перейти до нього;
- перевірити чи залишилися пункти, які не включені у маршрут, якщо результат позитивний – повторити крок 2;
- щоб завершити обхід додати ребро між останнім обраним пунктом та першим.

У загальному випадку трудомісткість рішення задачі дорівнює $O(n^2)$. Нижня межа вартості оптимального маршруту на 10% вище нижньої межі Хелд-Карп [17].

Жадібний алгоритм (Greedy). Щоб вирішити TSP використанням жадібного алгоритму, ми досліджуємо всі ребра, що виходять з пункта-вузла, і обираємо n найкоротших дуг. Якщо ті n найкоротших дуг формують гамільтонів цикл, тоді ми знайшли оптимальний розв'язок [18].

Трудомісткість виконання завдання жадібним алгоритмом дорівнює $O(n^2)$. Нижня межа вартості оптимального маршруту вище нижньої межі

Хелд-Карп на 15-20% [16].

Алгоритм Керігана – Ліна (Lin-Kerighan). Алгоритм Керніган - Ліна вважається одним із найбільш ефективних методів пошуку оптимальних або майже оптимальних розв’язків задачі комівояжера. Однак розробка і реалізація алгоритму дуже непроста, так як алгоритм складається з безлічі кроків, більшість з яких сильно впливає на роботу алгоритму [19]. Створення алгоритму Керніган було натхненне наглядом, що статичне К в оптимальному методі не дає найкращий розв’язок. З'явилася ідея використовувати різні стадії оптимального методу у виконанні евристичного алгоритму. На практиці було показано, що практично неможливо заздалегідь передбачити яке К слідує використовувати, щоб досягти кращого компромісу між трудомісткістю і якістю розв’язку. Лін і Керниган прибрали цей недолік, ввівши 21 оптимальну змінну, таким чином значення К змінюється під час виконання алгоритму [20]. Трудомісткість при цьому дорівнює $O(n^{2.2})$ [16].

Алгоритм пошуку із заборонами (TabuSearch). Головна проблема алгоритму найближчого сусіда полягає в частому застряванні в точці локального оптимуму. Цього можна уникнути, застосувавши алгоритм пошуку із заборонами, в 1977 році запропонований Ф. Гловером. Даний метод дозволяє переходити від одного локального оптимуму до іншого в пошуку глобального оптимуму, після переходу ребро потрапляє в список заборон і повторно не використовується, крім випадків, коли воно може поліпшити побудований оптимальний шлях. На практичному рівні заборонений набір зберігається як комбінація раніше відвідуваних кроків, який дозволяє побудувати подальший шлях щодо поточного рішення і сусідніх вузлів [21]. Головним недоліком цього методу є його час виконання - трудомісткість алгоритму оцінюється як $O(n^3)$ [16].

Мурашиний алгоритм (Ant Colony Optimization). Мурашиний алгоритм - ефективний поліноміальний алгоритм, натхненний поведінкою справжніх мурах. Вперше його принципи були описані в 1991 Марко Доріго.

Мурашкам властиво співпрацювати в пошуках харчових ресурсів, тому вони залишають слід хімічної речовини, феромонів, на їх шляху від гнізда до джерела їжі [20]. Цей тип невербальної комунікації називають стігмергія - стимуляція, заснована на досвіді попередніх мурах і спрямована на підвищення продуктивності [22].

Для вирішення завдання комівояжера як правило використовують близько 20 мурах. Їх розміщують у випадкові міста і відправляють в інші міста. Їм не дозволяють двічі відвідувати одне і те ж місто, тільки якщо вони не завершують маршрут. Той мураха, який обрав найкоротший тур, буде залишати слід феромонів обернено пропорційний довжині маршруту. Цей слід феромонів буде зчитуватися наступним мурахою при виборі міста, і з великою ймовірність він піде тим же шляхом, ще сильніше зміцнить слід. Цей процес буде багаторазово повторений поки не буде знайдено маршрут, досить короткий, щоб бути оптимальним.

Серед недоліків алгоритму хочеться виділити, що перший отриманий розв’язок може виявитися одним з найгірших в плані оптимізації, однак при повторному розв’язку метод видає досить точний результат [23].

Нижня межа Хелд-Карпа (The Held-Karp Lower Bound). Найпоширеніший спосіб виміряти ефективність евристичного алгоритму для вирішення TSP - це порівняти результати з нижньою межею Хелд-Карпа. Ця нижня межа є рішенням TSP, знайденим за поліноміальний час за допомогою симплекс-методу. Нижня межа Хелд-Карпа приблизно 0.8% нижче оптимальної тривалості туру [24]. У той же час вона гарантовано не перевищує оптимальний час більш ніж на 2/3 [16].

Станом на 2015 алгоритм Крістофідеса вважається найефективнішим методом для розв’язку завдачі комівояжера на загальних метричних просторах, хоча відомі кращі наближення для окремих випадків. Також добре в тестах себе показали алгоритм Керніган-Ліна і жадібний евристичний алгоритм [16].

3.4 Опис методів розв’язання

Для розв'язку своєї задачі я обрав алгоритм найближчого сусіда, оскільки він простий у реалізації, але щоб наблизити розв'язок до точного я використовую 3-Opt алгоритм, який дозволяє поліпшити результат, далі у розділі детальніше описуються методи обрані мною для розв'язання задачі про побудову найкоротшого шляху для маршруту, який містить багато проміжних пунктів.

Алгоритм найближчого сусіда, а також йому подібні інколи називають конструктивними, оскільки як їх часто використовують для побудови початкового маршруту, який може піддаватися подальшій оптимізації. Основними перевагами цього алгоритму є його відносно мала обчислювальна складність, простота розуміння і “природність” за принципом дії.

Кроки алгоритму:

- обрати довільну початкову точку;
- знайти найближчу невідвідану точку і перейти до неї;
- якщо лишаються невідвідані точки, то перейти до кроку 2;
- повернутися до початкової точки.

Обчислювальна складність алгоритму – $O(n^2)$. Якість отриманого розв'язку може бути гіршою від якості оптимального на 25–30 %, проте розв'язок буде отримано з мінімальними часовими затратами.

Після знаходження початкового шляху за допомогою алгоритма найближчого сусіда будемо оптимізовувати маршрут використовуючи 3-Opt

алгоритм. Алгоритм 3-Opt полягає у видаленні трьох ребер з вже існуючого шляху (отриманого за допомогою конструктивного алгоритму) і заміні їх трьома новими за умови, що буде отримано новий коротший шлях.

При видаленні ребра AB, CD та FE і заміною їх новими ребрами AD, BC та CF утвориться новий цикл, і якщо довжина нового шляху менша ніж у попереднього, то перебудовується існуючий шлях. У протилежному випадку аналізується наступне ребро, не змінюючи існуючого маршруту. Ця дія повторюється для кожної трійки ребер. Слід зазначити, що можна утворити дві можливі заміни, як показано на рисунку 3.1, інакше будуть утворюватися окремі цикли. На практиці цей алгоритм часто реалізовується як два або три застосування алгоритму 2-Opt. Отриманий у результаті оптимізації розв'язок буде гіршим від оптимального на 3-4% , обчислювальна складність алгоритму $O(n^3)$.

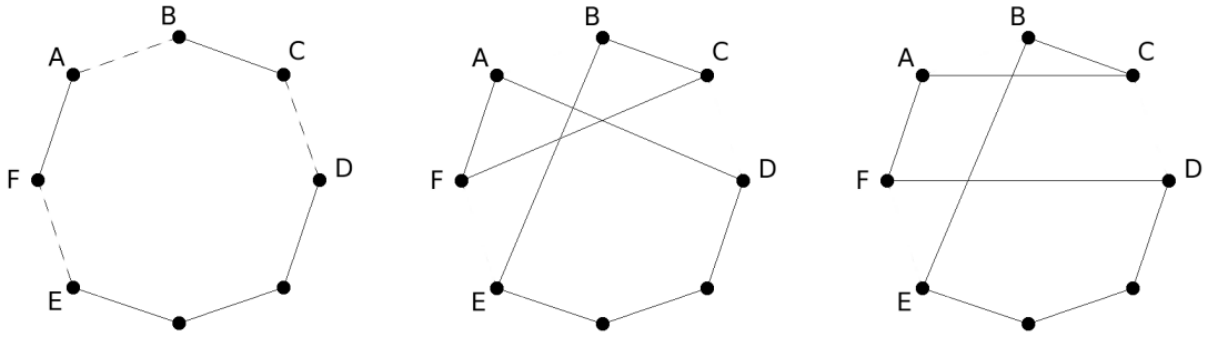


Рисунок 3.1 – Можливі перестановки при використанні алгоритму 3-Opt

Висновок до розділу

В даному розділі було сформульовано змістовну та математичну постановки задачі пошуку оптимального маршруту. Методом розв’язку було обрано алгоритм найближчого сусіда для пошуку початкового шляху, та 3-Opt алгоритм для оптимізації початкового шляху. Також було наведено детальний опис алгоритмів розв’язання поставленої задачі.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Для розробки мобільного застосування я обрав мову програмування Java, далі ми розглянемо їх переваги та недоліки і порівняємо з аналогами у таблиці 4.1.

Таблиця 4.1 – Порівняння мов програмування Java, Kotlin та Xamarin

Java(android)	Kotlin	Xamarin(C#)
Об'єктно орієнтована мова, що дає можливість створювати модульні програми, які дозволюють повторно використовувати вихідний код	Також є об'єктно орієнтованою мовою [2]	Також є об'єктно орієнтованою мовою
Належить до нативного типу розробки, а отже має повний доступ до вихідного коду платформи [4]	Також належить до нативного типу розробки [4]	Має обмежений доступ до вихідного коду платформи [4]
Оскільки Java це перша мова програмування, яка використовувалася для створення застосувань android, вона має дуже велику спільноту [3]	Оскільки Kotlin це нова мова програмування, то спільнота в неї дуже мала [3]	Оскільки Xamarin не користується великою популярністю серед розробників android, спільнота теж мала

Продовження таблиці 4.1

Для розробки потрібно знати лише нативні засоби розробки [1]	Також потрібно знати лише нативні засоби розробки [1]	Окрім стандартних засобів розробки Xamarin потрібно ще додатково знати нативні засоби розробки для всіх платформ, які використовуються [1]
Може використовуватись лише для мобільних застосувань на базі операційної системи android	Може використовуватись лише для мобільних застосувань на базі операційної системи android.	Є крос-платформеним засобом, тому має спільну бізнес-логіку для мобільних застосувань на базі операційних систем: android, IOS та Windows Phone [4]
Застосування є дуже простими та компактними [3]	Застосування з точки зору простоти та компактності поступаються Java [3]	Застосування з точки зору простоти та компактності поступаються Java та Kotlin [1]

Для зберігання даних я обрав базу даних Firebase Realtime Database, це база даних, розміщена у хмарі. Дані зберігаються як JSON і синхронізуються в реальному часі з кожним підключеним клієнтом. Основною перевагою для мене стало те, що цей сервіс безкоштовний і те, що незалежно від того на якій операційній системі працюють клієнти – вони усі мають справу з одним і тим самим сховищем даних.

Розглянемо основні переваги Firebase Realtime Database:

- замість використання HTTP запитів база даних Firebase Realtime Database використовує синхронізацію даних - кожен раз, коли дані змінюються, будь-який підключений пристрій отримує це оновлення протягом мілісекунд. Для моєї задачі швидкого обміну даними між керівником та перевізником цей варіант найкращий;
- база працює навіть у оффлайн режимі, тому що вона зберігає дані на диску, тобто на випадок, якщо клієнт втратив з'єднання база не зупинить свою роботу, а після відновлення з'єднання клієнт отримає будь-які зміни, які відбулися, поки він був у офлайн режимі, тобто він повністю синхронізується з поточним станом сервера;
- швидкість роботи – за рахунок постійної синхронізації з сервером, дані знаходяться у локальному сховищі, а тому не витрачається час на виконання запитів.

Із недоліків для себе можу виділити лише те, що я звик працювати з реляційними базами, де оперувати даними значно легше.

4.2 Вимоги до технічного забезпечення

Дане застосування реалізоване для мобільної операційної системи android версії від 5.0, отже з досвіду використання мобільних девайсів для користування застосуванням потрібен телефон з такими характеристиками:

- об'єм оперативної пам'яті від 2 гб;
- процесор потужністю від 1.6 ГГЦ 4 ядра;
- об'єм вільної пам'яті від 100 мб;
- операційна система Android 5.0 і вище;
- обов'язкове підключення до мережі інтернет.

4.3 Архітектура програмного забезпечення

Архітектура програмного забезпечення системи зображує організацію

або структуру системи і надає пояснення щодо її поведінки. Система являє собою набір компонентів, які виконують певну функцію або набір функцій. Іншими словами, архітектура програмного забезпечення забезпечує міцний фундамент, на якому можна будувати програмне забезпечення.

Для даного застосування була обрана хмарна архітектура, з базою даних у режимі реального часу. База даних знаходиться в хмарі, та всі пристрої, які під'єднані до неї постійно синхронізуються, будь – які дії користувача пов'язані зі зміною даних одразу ж оновлюють дані у інших користувачів. Оскільки використовуються хмарні технології, то немає жодної необхідності у серверах. Також хмарні технології надають можливість доступу до однієї і тієї ж бази даних користувачам, які використовують девайси на абсолютно різних платформах. На рисунку 4.1 зображено модель хмарної архітектури.

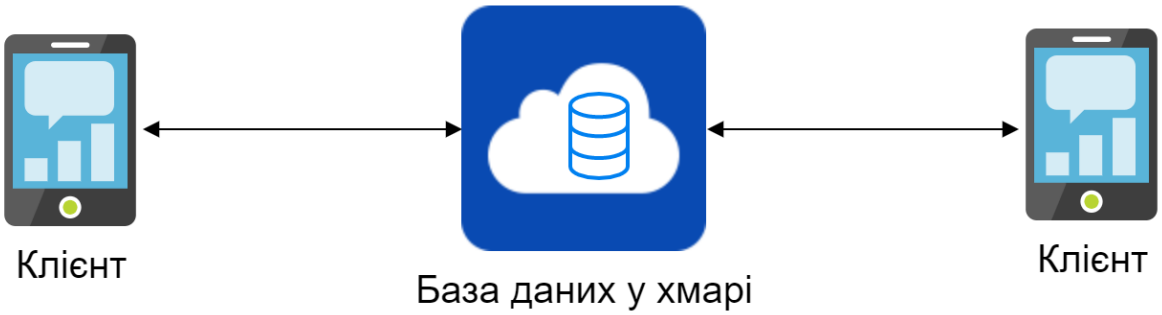


Рисунок 4.1 – Модель хмарної архітектури

4.3.1.1 Діаграма класів

Діаграми класів є основою об'єктно-орієнтованого аналізу та проектування. Діаграми класів показують класи системи, їх взаємозв'язки (включаючи успадкування, агрегацію та асоціацію), а також методи та поля класів. Діаграми класів використовуються для найрізноманітніших цілей, включаючи як концептуальне/доменне моделювання, так і детальне моделювання проектів.

В даному розділі було створено схему структурну класів, яка частково реалізує декілька шаблонів проектування, розглянемо їх.

FactoryMethod – породжуючий шаблон проектування, задача якого створювати об’єкти різних класів в залежності від поведінки програми одним методом, у нашому випадку цей шаблон реалізований у класі MapController, метод getMapController(User): MapController в залежності від типу користувача, який є аргументом цього методу, мета реалізації цього шаблону – розділити функціонал мапи між керівником та перевізником.

MVC (Model – View- Controller) – це архітектурний шаблон проектування, основна ідея якого полягає у розділенні логіки програми. Програма поділяється на такі 3 частини:

- model – клас, який відповідає моделі даних, з якою проводяться різні операції в ході роботи програми;
- view – клас, який відповідає за візуалізацію даних, які містяться у моделі;
- controller – клас, який діє на Model і на View, він контролює потік даних Model і оновлює View.

У нашій моделі шаблон MVC було реалізовано частиною класів, які відповідають за мапу, в ролі уявлення виступає клас MapView, у ролі моделі виступає клас MapState, у ролі контролера – MapController.

Repository – шаблон, завдання якого абстрагувати, основну логіку програми від операцій з базою даних, та бути єдиною ланкою між основною логікою програми, та базою даних. У нашому випадку шаблон репозиторій реалізований класом Repository, він одразу зчитує у кеш з бази даних як користувачів, так і усі маршрути, що дозволяє дуже просто проводити різні операції з даними.

Схема структурна класів наведена у графічному матеріалі.

4.3.2 Діаграма послідовності

Діаграма послідовності - це діаграма взаємодії, яка детально описує, як виконуються операції. Вони фіксують взаємодію між об'єктами в контексті співпраці. Діаграма послідовності є фокусом часу, і вона показує порядок взаємодії візуально, використовуючи вертикальну вісь діаграми для представлення часу, які повідомлення надсилаються і коли.

На схемі структурній послідовності зображений процес надсилання маршруту перевізнику, та процес відслідковування керівником поточного місця положення перевізника.

Надсилання маршруту перевізнику здійснюється записом цього маршруту у базу даних та прив’язуванням до нього перевізника. Отримання маршруту перевізником здійснюється зчитуванням з бази маршруту до якого він прив’язаний. Відслідковування поточного місця знаходження перевізника здійснюється зчитуванням з бази даних поточних координат перевізника на мапі, які з періодичністю перезаписуються до бази даних і так відбувається до тих пір, поки керівник не завершить перевезення натисненням на кнопку «Завершити перевезення». Схема структурна послідовності наведена у графічному матеріалі.

4.3.3 Діаграма компонентів

Діаграми компонентів відрізняються за характером і поведінкою. Діаграми компонентів використовуються для моделювання фізичних аспектів системи. Фізичними аспектами є такі елементи, як виконувані файли, бібліотеки, файли, документи і т.д., які знаходяться в вузлі.

Компонентні діаграми використовуються для візуалізації організації та відносин між компонентами системи. Ці діаграми також використовуються для створення виконуваних систем.

Як ми можемо побачити, поточна схема структурна компонентів складається з 4-х компонентів:

- графічний інтерфейс користувача - взаємодіє з основною логікою програми за допомогою Api користувача, відповідальна за відображення користувачу даних;
- логіка обчислень - бере на себе куревання усім проектом, вона взаємодіє з сервісам google maps, за допомогою google maps Api та взаємодіє з хмарним сервісом бази даних за допомогою Firebase Api;
- хмарний сервіс бази даних – хмарний сервіс, через який відбувається взаємодія з базою даних, яка розгорнута у хмарі, відповідальний за переміщення даних між логікою обчислень та хмарою;
- сервіс мапи – сервіс мап , який відповідальний за отримання та передачі інформації на мапу.

Схема структурна компонентів наведена у графічному матеріалі.

4.3.4 Специфікація функцій

Для даної предметної області було створено методи для класів, які зображені на схемі структурній класів.

В таблицях 4.2 - 4.12 можна побачити опис відкритих методів класів, які були створені для реалізації даного дипломного проекту. Опис має таку структуру:

- назва методу;
- опис призначення методу;
- перелік аргументів, які метод приймає на вході, якщо вони є;
- опис вхідних аргументів методу.

Таблиця 4.2 – Методи класу MapView

Метод	Опис методу
decodePath	Розкодовує побудований маршрут у тип даних, для зручної роботи з мапою. Сам маршрут зберігається у полі цього класу data
encodePath	Закодовує побудований маршрут у тип даних, для зручного зберігання у сховищу данних

Таблиця 4.3 – Методи класу MapController

Метод	Опис методу	Параметр	Опис параметру
showMap	Відображає мапу для користувачів	–	–
getPosition	Повертає значення поточного місця маркера	–	–
getMapController	Повертає екземпляр класу DirectorMapController чи DriverMapController в звлежності від вхідних аргументів	user	Екземпляр будь-якого класу, який наслідує клас User

Таблиця 4.4 – Методи класу DirectorMapController

Метод	Опис методу	Параметр	Опис параметру
buildRoute	Будує та оптимізує маршрут за точками вказаними у полі resultRoute цього ж класу, результат повертає у це ж саме поле	–	–
addMarker	Додає маркер на мапу, та записує його координати у поле цього ж класу resultRoute	dot	Екземпляр класу Dot – точка цього маркера на мапі

Продовження таблиці 4.4

Метод	Опис методу	Параметр	Опис параметру
removeMarker	Видаляє маркер з мапи, та поля resultRoute	dot	Екземпляр класу Dot – точка, яку необхідно видалити
sendRouteToDriver	Надсилає побудований керівником маршрут перевізнику	id	Числовий ідентифікатор користувача - перевізника
showDriver	Відображає координати, в яких знаходиться перевізник на мапі	id	Числовий ідентифікатор користувача - перевізника

Таблиця 4.5 – Методи класу DriverMapController

Метод	Опис методу
showMap	Відображає мапу з маршрутом для перевізника

Таблиця 4.6 – Методи класу RouteBuilder

Метод	Опис методу	Параметр	Опис параметру
buildRoute	Будує початковий маршрут, по точкам, які приходять із параметрів, та оптимізує його	route	Екземпляр класу Route – виступає у ролі колекції точок, за якими потрібно будувати маршрут

Таблиця 4.7 – Методи класу NearestNeighborAlgorithm

Метод	Опис методу	Параметр	Опис параметру
exec	Будує початковий маршрут, по точкам, які приходять із параметрів, реалізуючи алгоритм найближчого сусіда	route	Екземпляр класу Route – виступає у ролі колекції точок, за якими потрібно будувати маршрут

Таблиця 4.8 – Методи класу ThreeOptAlgorithm

Метод	Опис методу	Параметр	Опис параметру
optimize	Приймає на вхід початковий маршрут, побудований алгоритмом найближчого сусіда, та оптимізує його використовуючи 3-Opt алгоритм.	route	Екземпляр класу Route – виступає у ролі початково побудованого маршруту

Таблиця 4.9 – Методи класу TwoOptAlgorithm

Метод	Опис методу	Параметр	Опис параметру
optimize	Приймає на вхід початковий маршрут, побудований алгоритмом найближчого сусіда, та оптимізує його використовуючи 2-Opt алгоритм.	route	Екземпляр класу Route – виступає у ролі початково побудованого маршруту

Таблиця 4.10 – Методи класу Repository

Метод	Опис методу	Параметр	Опис параметру
retrieveUsers	Зчитує інформацію про усіх користувачів підприємства з бази даних у кеш	–	–
retrieveRoute	Зчитує маршрути пов’язані з користувачем з бази даних у кеш	idUser	Числовий ідентифікатор користувача
addUser	Реєструє у системі нового користувача, записуючи всі його дані у базу даних	user	Екземпляр класу User, виступає у ролі представлення всієї інформації користувача в оперативній пам’яті
sendRoute	Надсилає маршрут перевізнику шляхом прив’язування до нього перевізника у базі даних	idUser	Числовий ідентифікатор користувача
deleteUser	Видаляє аккаунт користувача з бази даних	idUser	Числовий ідентифікатор користувача
getUser	Зчитує з кеша, або бази даних усю інформацію користувача, за його логіном	login	Логін користувача
deleteRoute	Видалення маршруту з бази даних	route	Екземпляр класу Route – уся інформація про маршрут

Таблиця 4.11 – Методи класу FirebaseDatabase

Метод	Опис методу	Параметр	Опис параметру
getInstance	Повертає екземпляр класу FirebaseDatabase, для можливості працювати з базою даних у хмарі	–	–
getReference	Повертає посилання на об’єкт бази даних у хмарі за її адресою, для можливості взаємодіяти з базою даних	path	Шлях до бази даних у хмарі

Таблиця 4.12 – Методи класу DataBaseReference

Метод	Опис методу	Параметр	Опис параметру
orderByChild	Створює запит до бази даних у хмарі, задає порядок сортування	child	Рядок – назва наслідника у ієрархічному списку БД
startAt	Створює чи доповнює запит до бази даних у хмарі, задає фільтр на початкове значення поля з колекції у базі даних	childStart	Субдані поля, початкова частина, за типом даних довільний об’єкт
endAt	Створює чи доповнює запит до бази даних у хмарі, задає фільтр на кінцеве значення поля з колекції у базі даних	childEnd	Субдані поля, кінцева частина, за типом даних довільний об’єкт
setValue	Записати нові чи оновлені дані у базу даних у хмарі	dbRef	Посилання на об’єкт бази даних у хмарі
		listener	Подія сповіщена, коли операція підтверджена сервером бази даних

Продовження таблиці 4.12

Метод	Опис методу	Параметр	Опис параметру
updateChildren	Записати оновлені дані в деякі поля у базі даних на хмарі	dictionary	Словник, ключем у якому є назва поля, а значенням нове значення цього поля
removeValue	Видалити дані з бази даних у хмарі	dbRef	Посилання на об’єкт бази даних у хмарі
		listener	Подія сповіщена, коли операція підтверджена сервером бази даних

Висновок до розділу

У цьому розділі були порівняні та описані засоби розробки, які були використані для реалізації застосування з підтримки логістичної діяльності малих підприємств.

Визначені вимоги до технічного забезпечення, необхідного для коректного функціонування застосування.

Представлена архітектура програмного забезпечення, у даному випадку клієнт-сервер з використанням хмарних технологій.

Наведено детальний опис схеми структурної класів та схеми структурної послідовності, що демонструють наявні в застосуванні класи та порядок їх взаємодії із базою даних та користувачем.

За допомогою схеми структурної компонентів показане розбиття програмної частини застосування на структурні компоненти та їх залежність один з одним.

Наведена специфікація методів, які були використані у застосуванні.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

При відкритті застосування користувач потрапляє на сторінку авторизації, де він бачить форму для заповнення логіну та пароля, а також дві кнопки «Вхід» та «Реєстрація», якщо користувач вже зареєстрований, то він може авторизуватися ввівши логін, пароль та натиснувши кнопку «Вхід».

Якщо користувач не зареєстрований у системі – він має зареєструватися, натиснувши на кнопку «Реєстрація». Натиснувши на цю кнопку користувач потрапить на сторінку реєстрації, де йому буде необхідно заповнити особисті дані.

Після авторизації користувач потрапляє на сторінку головного меню, ця сторінка має різний вигляд для різних типів користувачів. На головній сторінці перевізника можливо лише перейти до перегляду мапи з маршрутом, а на головній сторінці керівника можливо перейти до сторінки побудови маршрута, перейти до сторінки редагування перевізників, та перейти до сторінки контролю дотримання маршрута.

Зі сторінки головного меню керівник може перейти до сторінки редагування перевізників, там він може додавати, чи звільняти нових перевізників.

Наступною доступною керівнику дією є перехід до сторінки побудови маршрутів, на цій сторінці керівник може побудувати маршрут на мапі, для цього йому необхідно відмітити точки маршруту на мапі, для того щоб поставити точку, йому просто необхідно обрати та затримати точку на мапі, після чого на цій точці з'явиться маркер.

Для побудови маршрута після того, як користувач обрав усі необхідні йому точки маршрута – йому необхідно натиснути на кнопку з малюнком гайкового ключа, яка знаходиться у нижньому лівому куті, після натискання на цю кнопку зникнуть усі маркери, та відобразиться маршрут.

Також при побудові маршруту керівник може користуватися пошуком на мапі, для більш зручного прокладання маршрутів, для цього йому необхідно просто набрати у верхньому полі пошуку дані про необхідну локацію.

Наступна і остання функція, яка доступна керівнику – це контроль дотримання маршруту. Перейшовши з головного меню до цієї функції користувачу необхідно спершу обрати перевізника, якого він бажає контролювати. Після вибору перевізника зі списку він потрапляє на сторінку з мапою, на якій зображений маршрут, та поточне місце положення перевізника. Поточне місце положення перевізника на маршруті представлене маркером синього кольору.

Сторінки авторизації користувача, реєстрації користувача, головного меню користувача, редагування перевізників, мапи для побудови маршруту з пошуковим рядком, мапи для побудови маршруту з маркерами, по яким будується маршрут, сторінка мапи для побудови маршруту з прокладеним маршрутом, мапи для контролю дотримання маршруту з місцем знаходження перевізника наведені у графічному матеріалі.

5.2 Випробування програмного продукту

У цьому підрозділі наведено опис тестових сценаріїв для перевірки відповідності застосування функціональним вимогам, представленим у технічному завданні.

5.2.1 Мета випробувань

Метою випробувань є перевірка правильності побудови маршруту, перевірка функціоналу відслідковування дотримання маршруту та перевірка відповідності функціоналу вимогам технічного завдання.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603−92. Інформаційна технологія. Види випробувань автоматизованих систем;

– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

В таблицях 5.1, 5.2, 5.3, 5.4, 5.6 наведені тест-кейси для контролю введення персональних даних користувача та пунктів маршруту.

Таблиця 5.1 – Тест реєстрації нового керівника

Функція/ Use Case:	Реєстрація	
Дія:	Очікуваний результат:	Результат тесту:
Передумова		
Відкрийте доданок	Відкрита сторінка авторизації	пройдений
Натисніть на кнопку «Реєстрація»	Відкрита сторінка реєстрації	пройдений
Кроки тесту		
Заповніть форму реєстрації Логін: «GBV» Пароль: «password» Електронна пошта: «bodya301097@gmail.com» По-батькові: «Віталійович» Ім’я: «Богдан» Прізвище: «Голубець» Телефон: «+380734169923» Підприємство: «KPI»	Дані успішно введені	пройдений
Натисніть кнопку «Зареєструватися»	Відкрита та доступна вкладка авторизації користувача	пройдений

Таблиця 5.2 – Тест авторизації користувача

Функція/ Use Case:	Авторизація	
Дія:	Очікуваний результат:	Результат тесту:
Передумова		
Відкрийте доданок	Відкрита сторінка авторизації	пройдений
Кроки тесту		
Заповніть форму авторизації Логін: «GBV» Пароль: «password»	Дані успішно введені	пройдений
Натисніть кнопку «Вхід»	Відкрита сторінка з головним меню	пройдений
Післяумова		
Натисніть на кнопку «Вихід з програми»	Відкрита сторінка авторизації	пройдений

Таблиця 5.3 – Тест реєстрації перевізника

Функція/ Use Case:	Реєстрація перевізника	
Дія:	Очікуваний результат:	Результат тесту:
Передумова		
Відкрийте доданок	Відкрита сторінка авторизації	пройдений
Заповніть форму авторизації Логін: «GBV» Пароль: «password»	Дані успішно введені	пройдений
Натисніть на кнопку «Вхід»	Відкрита сторінка з головним меню	пройдений
Натисніть на кнопку «Редагування перевізників»	Відкрита сторінка редагування перевізників	пройдений
Кроки тесту		
Натисніть на кнопку з рисунком + у правому верхньому куті	Відкрита сторінка реєстрації	пройдений

Продовження таблиці 5.3

Кроки тесту		
Заповніть форму реєстрації Логін: «Driver1» Пароль: «password» Електронна пошта: «drive@gmail.com» По-батькові: «Андрійович» Ім’я: «Петро» Прізвище: «Іванов» Телефон: «+380732153952» Підприємство: «KPI»	Дані успішно введені	пройдений
Натисніть на кнопку «Зареєструвати»	Відкрита сторінка редагування перевізників, серед списку перевізників присутній щойно зареєстрований перевізник	пройдений
Післяумова		
Натисніть на клавішу «Повернутися»	Відкрита сторінка з головним меню	пройдений

Таблиця 5.4 – Тест побудови маршруту

Функція/ Use Case:	Побудова маршруту	
Дія:	Очікуваний результат:	Результат тесту:
Передумова		
Відкрийте доданок	Відкрита сторінка авторизації	пройдений
Заповніть форму авторизації Логін: «GBV» Пароль: «password»	Дані успішно введені	пройдений
Натисніть на кнопку «Вхід»	Відкрита сторінка з головним меню	пройдений
Натисніть на кнопку «Побудувати маршрут»	Відкрита сторінка з мапою для побудови маршруту	пройдений
Кроки тесту		
Оберіть точку на мапі, та затисніть її на 1 секунду	На мапу додався маркер	пройдений

Продовження таблиці 5.4

Кроки тесту		
Додайте ще декілька маркерів	На мапі відображено декілька маркерів	пройдений
Натисніть на кнопку із зображенням гайкового ключа у лівому нижньому куті	З мапи зникли маркери, та відобразився побудований за ними маршрут	пройдений
Післяумова		
Натисніть на клавішу «Повернутися»	Відкрита сторінка з головним меню	пройдений

Таблиця 5.5 – Тест надсилання маршруту перевізнику

Функція/ Use Case:	Побудова маршруту	
Дія:	Очікуваний результат:	Результат тесту:
Передумова		
Відкрийте доданок	Відкрита сторінка авторизації	пройдений
Заповніть форму авторизації Логін: «GBV» Пароль: «password»	Дані успішно введені	пройдений
Натисніть на кнопку «Вхід»	Відкрита сторінка з головним меню	пройдений
Натисніть на кнопку «Побудувати маршрут»	Відкрита сторінка з мапою для побудови маршруту	пройдений
Оберіть точку на мапі, та затисніть її на 1 секунду	На мапу додався маркер	пройдений
Додайте ще декілька маркерів	На мапі відображено декілька маркерів	пройдений
Натисніть на кнопку із зображенням гайкового ключа у лівому нижньому куті	З мапи зникли маркери, та відобразився побудований за ними маршрут	пройдений
Кроки тесту		
Натисніть на кнопку із зображенням паперового літака у нижньому правому куті	Відкрита сторінка вибору перевізника	пройдений

Продовження таблиці 5.5

Кроки тесту		
Оберіть перевізника з прізвищем «Іванов», Ім’ям «Петро» зі списку та натисніть на нього	На екрані з’явилося повідомлення про успішне надсилання маршрута перевізнику, відкрилася сторінка головного меню	пройдений
Натисніть кнопку «Вихід з програми»	Відкрита сторінка авторизації	пройдений
Заповніть форму авторизації Логін: «Driver1» Пароль: «password»	Дані успішно введені	пройдений
Натисніть на кнопку «Вхід»	Відкрита сторінка з головним меню перевізника	пройдений
Натисніть на кнопку «Переглянути маршрут»	Відкрита сторінка з мапою, на якій зображений маршрут побудований під логіном керівника	пройдений
Післяумова		
Натисніть на клавішу «Повернутися»	Відкрита сторінка з головним меню	пройдений

Таблиця 5.6 – Тест контролю дотримання маршруту

Функція/ Use Case:	Контроль перевезень	
Дія:	Очікуваний результат:	Результат тесту:
Передумова		
Відкрийте доданок	Відкрита сторінка авторизації	пройдений
Заповніть форму авторизації Логін: «GBV» Пароль: «password»	Дані успішно введені	пройдений
Натисніть на кнопку «Вхід»	Відкрита сторінка з головним меню	пройдений
Натисніть на кнопку «Побудувати маршрут»	Відкрита сторінка з мапою для побудови маршруту	пройдений
Оберіть точку на мапі, та затисніть її на 1 секунду	На мапу додався маркер	пройдений

Продовження таблиці 5.6

Передумова		
Додайте ще декілька маркерів	На мапі відображено декілька маркерів	пройдений
Натисніть на кнопку із зображенням гайкового ключа у лівому нижньому куті	З мапи зникли маркери, та відобразився побудований за ними маршрут	пройдений
Натисніть на кнопку із зображенням паперового літака у нижньому правому куті	Відкрита сторінка вибору перевізника	пройдений
Оберіть перевізника з прізвищем «Іванов», Ім’ям «Петро» зі списку та натисніть на нього	На екрані з’явилося повідомлення про успішне надсилання маршрута перевізнику, відкрилася сторінка головного меню	пройдений
Кроки тесту		
Натисніть на кнопку «Контроль дотримання маршруту»	Відкрита сторінка зі списком перевізників	пройдений
Оберіть перевізника з прізвищем «Іванов», Ім’ям «Петро» зі списку та натисніть на нього	Відкрита сторінка з мапою, на якій зображений маршрут, та поточне місце знаходження перевізника	пройдений
Післяумова		
Натисніть на клавішу «Повернутися»	Відкрита сторінка з головним меню	пройдений

Висновок до розділу

В розділі було наведено детальну таблицю тест-кейсу, яка покриває функціонал введення персональних даних користувача, та перевірку функціонування системи.

ЗАГАЛЬНІ ВИСНОВКИ

У дипломній роботі було розглянуто основні задачі для оптимізації роботи малих підприємств, шляхом створення застосування для підтримки логістичної діяльності малих підприємств.

У розділ загальних положень було надано інформації про предметне середовище роботи. Також було описано акторів – керівника, перевізника та їх функції. Було проведено огляд наявних аналогів, у якому було розглянуто їх основні функції та можливості з вказанням їх недоліків.

У розділі інформаційного забезпечення було детально описано структуру сховища даних, вхідних та вихідних даних.

У розділі програмної та технічної підтримки було описано техноло-гії, які використовувались при розробці функціоналу застосування. В якості сховища даних було обрано хмарну технологію Firebase Realtime Data Base, інструментом розробки було обрано мову програмування Java, а засобом роботи з мапами було обрано Google maps Api Android Sdk. Для нормальної роботи функціоналу, було наведено вимоги до технічного та програмного забезпечень.

У технологічному розділі було наведено керівництво користувача, яке містить детальні інструкції користувачу про те, як користуватись розробленим функціоналом. Також було описано методи випробування на основі тестових сценаріїв.

Розроблений функціонал «EasyRoute» розширив деякий функціонал інших аналогів завдяки можливості відслідковування дотримання маршруту у режимі реального часу.

ПЕРЕЛІК ПОСИЛАНЬ

1. https://wellsoft.pro/blog/sravnenie-xamarin-s-nativnymiios-android-i-gibridnymirazrabotkami
2. https://dou.ua/lenta/articles/java-vs-kotlin/
3. https://ua-blog.com/kotlin-vs-java-%D0%BD%D0%B0-%D1%87%D0%B5%D0%BC-%D0%BF%D0%B8%D1%81%D0%B0%D1%82%D1%8C-%D0%BF%D0%BE%D0%B4-android/
4. https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/
5. Логіст. http://logist.poncy.ru/
6. Speedy Route. https://www.speedyroute.com/
7. Левитин А. Алгоритмы. Введение в разработку и анализ. Вильямс, 2006. 35–36 с.
8. Kona H., Burde A., Dr. Zanwar D. R. A Review of Traveling Salesman Problem with Time Window Constraint // IJIRST – International Journal for Innovative Research in Science & Technology, 2015. Vol. 2, Issue 1. P. 253–254.
9. Tannenbaum P. Excursions in Mathematics. University Kansas, 2011. P. 25.
10. Clausen J. Branch and Bound Algorithms – Principles and Examples. University of Copenhagen, 1999. P. 5-6.
11. Tollis I. G. Algorithms and Complexity. University of Crete, 2000. P. 140–146.
12. Applegate D., Bixby R., Chvatal V., Cook W. TSP cuts outside the template paradigm. Donet, 2000. P. 1–10.
13. Захарова Е.М., Минашина И.К. Обзор методов многомерной оптимизации // Информационные процессы, 2014. Том 14, № 3 стр. 265–266.
14. Papadimitriou C., Vazirani U. Efficient Algorithms and Intractable Problems.
15. Goodrich M., Roberto T. Algorithm Design and Applications, Wiley, 2015. P.

16. Nilsson C. Heuristics for the Traveling Salesman Problem. Linkoping University, 2011. P. 1–6.
17. Alsalibi B.A., Jelodar M.B., Venkat I. A Comparative Study between the Nearest Neighbor and Genetic Algorithms: A revisit to the Traveling Salesman Problem // International Journal of Computer Science and Electronics Engineering (IJCSEE), 2013. Vol. 1, Issue 1.
18. Gutin G.,Yeo A. The Greedy Algorithm for the Symmetric TSP. University of London, 2002. P. 1–2.
19. Gutin G., Karapetyan D. Lin-Kernighan Heuristic Adaptations for the Generalized Traveling Salesman Problem. Royal Holloway London University, 2010. P. 1–5.
20. Gupta R., Chauhan C., Pathak K. Survey of Methods of Solving TSP along with its Implementation using Dynamic Programming Approach. // International Journal of Computer Applications, 2012. Vol. 52, No.4. P. 1–6.
21. Basu S. Tabu Search Implementation on Traveling Salesman Problem and Its Variations: A Literature Survey. Indian Institute of Management Calcutta, 2012. P. 1–8.
22. Dorigo M., Caro G. D. Ant Algorithms for Optimization // Artificial Life, 1999. Vol. 5, No 2. P. 139–140.
23. Dorigo M., Gambardella L. M. Ant colonies for the traveling salesman problem. Université Libre de Bruxelles, 1996. P. 1-4.
24. Johnson D. S., McGeoch L. A., Rothberg E. E. Experimental Analysis for the Held-Karp Traveling Salesman Bound. AT&T Bell Laboratories, 1999. P. 1–

Додаток А

Тексти програмного коду

Комплекс задач підтримки логістичної діяльності малих підприємств

(Найменування програми (документа))

DVD-R

(Вид носія даних)

7 арк, 48800 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2019 року

```
class ThreeOptAlgorithm
{

          public Route optimize(Route route)
          {
                    float distance = route.getLength();

                    for (int i = 1; i < route.getDotCount()-3; ++i)
                    {
                              for (int j = i+1; j < route.getDotCount()-2; ++j)
                              {
                                        for (int k = j+1; k < route.getDotCount()-1; ++k)
                                        {
                                                  // Perform the 3 way swap and test the length
                                                  route.swapIndexes(i, k);
                                                  route.swapIndexes(j, k);
                                                  float newDistance = route.getLength();

                                                  if (newDistance < distance)
                                                            return optimize(route);
                                                  else
                                                  {
                                                            route.swapIndexes(j, k);
                                                            route.swapIndexes(i, k);
                                                  }
                                        }
                              }
                    }

                    return route;
          }
}
```

```
        import java.util.*;
        import java.lang.*;
        import java.io.*;

        class Dot {
          double x;
          double y;

          public Dot(double x1, double y1) {
            x=x1;
            y=y1;
          }

        }

        class CompareByX implements Comparator<Dot> {
            public int compare(Dot p1, Dot p2) {
            if (p1.x < p2.x) return -1;
                if (p1.x == p2.x) return 0;
                return 1;
            }
        }
```

```
    /* An object of the above comparator class is used by java.util.Arrays.sort() in
main to sort an array of points by x-coordinates */

class Auxiliaries {

    public static double distSquared(Dot p1, Dot p2) {
        double result;
        result = (p1.x-p2.x)*(p1.x-p2.x) + (p1.y-p2.y)*(p1.y-p2.y);
        return result;
    }

}

public class NearestNeighborsAlgorithm{
    public static void exec (Route route) throws IOException {
        int range = 1000000; // Range of x and y coordinates in points

        String npoints = buffer1.readLine();
        int numpoints = Integer.parseInt(npoints);

        // numpoints is now the number of points we wish to generate

        Dot inputpoints [] = new Dot [numpoints];

        // array to hold points
        int closest [] = new int [numpoints];
        // array to record soln; closest[i] is index of point closest to i'th
        int px, py;
        double dx, dy, dist;
        int i,j;

        double currbest;
        int closestPointIndex;
        long tStart, tEnd;


        for (i = 0; i < numpoints; i++) {

          px = (int) ( range * Math.random());
          dx = (double) px;
          py = (int) (range * Math.random());
          dy = (double) py;
          inputpoints[i] = new Dot(dx, dy);
```

```
}

// array inputpoints has now been filled



tStart = System.currentTimeMillis();

// find closest [0]


closest[0] = 1;



currbest = Auxiliaries.distSquared(inputpoints[0],inputpoints[1]);
for (j = 2; j < numpoints; j++) {
   dist = Auxiliaries.distSquared(inputpoints[0],inputpoints[j]);
   if (dist < currbest) {
       closest[0] = j;
       currbest = dist;
   }
}

// now find closest[i] for every other i

for (i = 1; i < numpoints; i++) {
    closest[i] = 0;
    currbest = Auxiliaries.distSquared(inputpoints[i],inputpoints[0]);
    for (j = 1; j < i; j++) {
      dist = Auxiliaries.distSquared(inputpoints[i],inputpoints[j]);
      if (dist < currbest) {
       closest[i] = j;
       currbest = dist;
  }

    }
    for (j = i+1; j < numpoints; j++) {
      dist = Auxiliaries.distSquared(inputpoints[i],inputpoints[j]);
      if (dist < currbest) {
  closest[i] = j;
         currbest = dist;
  }
    }
}
```

```
            }
        }
public class TwoOpt {
                            public static ArrayList<Point2D> alternate(ArrayList<Point2D> ci-
                        ties) {
                                ArrayList<Point2D> newTour;
                                double bestDist = Length.routeLength(cities);
                                double newDist;
                                int swaps = 1;
                                int improve = 0;
                                int iterations = 0;
                                long comparisons = 0;


                                while (swaps != 0) { //loop until no improvements are made.
                                    swaps = 0;


                                    //initialise inner/outer loops avoiding adjacent calcula-
                        tions and making use of problem symmetry to half total comparisons.




            for (int i = 1; i < cities.size() - 2; i++) {
                  for (int j = i + 1; j < cities.size() - 1; j++) {
                      comparisons++;
                      //check distance of line A,B + line C,D against A,C + B,D if there is
    improvement, call swap method.
                      if ((cities.get(i).distance(cities.get(i - 1)) + cities.get(j +
    1).distance(cities.get(j))) >=
                                  (cities.get(i).distance(cities.get(j + 1)) + cities.get(i -
    1).distance(cities.get(j)))) {


                          newTour = swap(cities, i, j); //pass arraylist and 2 points to be
    swapped.

                          newDist = Length.routeLength(newTour);


                          if (newDist < bestDist) { //if the swap results in an improved dis-
    tance, increment counters and update distance/tour
                                cities = newTour;
                                bestDist = newDist;
                                swaps++;
```

```
                        improve++;
                    }
                }
            }
        }
        iterations++;
    }
    System.out.println("Total comparisons made: " + comparisons);
    System.out.println("Total improvements made: " + improve);
    System.out.println("Total iterations made: " + iterations);
    return cities;
}


private static ArrayList<Point2D> swap(ArrayList<Point2D> cities, int i, int j) {
    //conducts a 2 opt swap by inverting the order of the points between i and j
    ArrayList<Point2D> newTour = new ArrayList<>();


    //take array up to first point i and add to newTour
    int size = cities.size();
    for (int c = 0; c <= i - 1; c++) {
        newTour.add(cities.get(c));
    }


    //invert order between 2 passed points i and j and add to newTour
    int dec = 0;
    for (int c = i; c <= j; c++) {

        newTour.add(cities.get(j - dec));
        dec++;
    }


    //append array from point j to end to newTour
    for (int c = j + 1; c < size; c++) {
        newTour.add(cities.get(c));
    }


    return newTour;
}
}

 private List<GeocodedWaypoint> geocodedWaypointList;
 @SerializedName("routes")
```

```
private List<Route> routeList;
@SerializedName("status")
private String status;
@SerializedName("error_message")
private String errorMessage;


public Direction() {
}


protected Direction(Parcel in) {
    status = in.readString();
    errorMessage = in.readString();
}


public void setGeocodedWaypointList(List<GeocodedWaypoint> geocodedWaypointList) {
    this.geocodedWaypointList = geocodedWaypointList;
}


public List<GeocodedWaypoint> getGeocodedWaypointList() {
    return geocodedWaypointList;
}


public void setRouteList(List<Route> routeList) {
    this.routeList = routeList;
}


public List<Route> getRouteList() {
    return routeList;
}


public void setStatus(String status) {
    this.status = status;
}


public String getStatus() {
    return status;
}
```

```
    public void setErrorMessage(String errorMessage) {
        this.errorMessage = errorMessage;
    }


    public String getErrorMessage() {
        return errorMessage;
    }


    public boolean isOK() {
        return RequestResult.OK.equals(status);
    }


    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(status);
        dest.writeString(errorMessage);
    }


    @Override
    public int describeContents() {
        return 0;
    }


    public static final Creator<Direction> CREATOR = new Creator<Direction>() {
        @Override
        public Direction createFromParcel(Parcel in) {
            return new Direction(in);
        }


        @Override
        public Direction[] newArray(int size) {
            return new Direction[size];
        }
    };
}
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО
Керівник проекту
_____ *О.В. Ковтунець*
(підпис) (ініціали, прізвище)
“16” травня 2019 р.

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)
“17” травня 2019 р.

Комплекс задач підтримки логістичної діяльності
малих підприємств

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП ІС-5206.1181-с.ТЗ*

на 9 сторінках

Київ – 2019 року

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Повна назва системи: комплекс задач підтримки логістичної діяльності малих підприємств.

Коротке найменування системи: «EasyRoute».

1.2 Найменування організації-замовника на організацій-учасників робіт

Генеральним замовником проекту являється кафедра Автоматизованих систем обробки інформації та управління НТУУ «КПІ». Представником замовника є Олесь Володимирович Ковтунець.

Розробником застосунку є студент групи ІС-52 факультету інформатики та обчислювальної техніки НТУУ «КПИ ім. Ігоря Сікорського» Голубець Богдан.

1.3 Перелік документів, на підставі яких створюється система

При розробці системи і створення проектно-експлуатаційної документації

Виконавець повинен керуватися вимогами наступних нормативних документів:

- ДСТУ 19.201-78. Технічне завдання. Вимоги до змісту і оформлення;
- ДСТУ 34.601-90. Комплекс стандартів на автоматизовані системи;
- ДСТУ 34.201-89. Інформаційні технології. Комплекс стандартів на автоматизовані системи. Види, комплексність і позначення документів при створенні автоматизованих систем.

1.4 Планові терміни початку і закінчення роботи зі створення застосунку

Плановий термін початку роботи над створенням застосунку – 5 лютого 2019 року.

Плановий термін по закінченню роботи над створенням застосунку – не пізніше 1 червня 2019 року.

2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ ЗАСТОСУНКУ

2.1 Призначення застосунку

Застосунок призначений для полегшення роботи з логістикою, а саме планування маршруту за допомогою графічного застосунку.

2.2 Цілі створення застосунку

Цілями розробки є:

- легка робота з маршрутами;
- оптимізація маршрутів;
- контроль дотримання маршруту.

Для досягнення поставлених цілей необхідно розв’язати наступні задачі:

- реалізувати реєстрацію та авторизацію користувачів;
- інтегрувати карту у систему;
- реалізувати алгоритм оптимізації маршруту;
- реалізувати збереження історії маршрутів у базу даних;
- розробити повідомлення про відхилення від маршруту.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Для користування застосуванням, користувач повинен зареєструватись та авторизуватись в системі.

Об'єктом автоматизації є операції оптимізації маршруту переміщення та відслідковування вантажу.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Застосунок має виступати в ролі планувальника перевезень для керівника та контролера перевізника.

Застосунок має виконувати наступні функції:

- реалізувати реєстрацію та авторизацію користувачів;
- планування маршруту перевезень;
- контроль перевізника, шляхом відстеження дотримання маршруту.

4.2 Вимоги до надійності

Застосунок повинен безвідмовно функціонувати, не зважаючи ймовірні дефекти, які можуть проявлятися під час експлуатації. Виправлення виявлених дефектів повинно здійснюватися на етапах попередніх випробувань.

Відмова програмного забезпечення застосунку, або відсутність підключення до мережі інтернет не повинна викликати пошкодження даних у інформаційному сховищі, у разі перебоїв з'єднання з мережею інтернет застосунок повинен автоматично поновити коректну роботу після відновлення з'єднання з мережею.

4.4 Вимоги до складу і параметрів технічних засобів

Для коректної роботи даного продукту на стороні користувача мають бути такі технічні характеристики:

- смартфон:
  1) процесор з частотою - не менше 2 ГГц, 4 ядра;
  2) об'єм оперативної пам'яті - не менше 2 ГБ.
- програмне забезпечення:
  1) операційна система Android 5.2+;
  2) підключення до мережі інтернет.

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Основні етапи виконання робіт з розробки системи ведення наукової роботи.

№ п/п	Назва етапу роботи	Термін виконання етапу	Результат виконання
1.	Підготовка технічного завдання на розробку програмного продукту	07.02.2019	
2.	Створення архітектури системи	12.02.2019	
3.	Технічне проектування – функціональність, модулі, задачі, цілі тощо	20.02.2019	
4.	Узгодження з керівником інтерфейсу користувача	02.03.2019	
5.	Розробка програми	10.04.2019	
6.	Налагодження програми	20.04.2019	
7.	Тестування програми	16.05.2019	
8.	Здача готового програмного продукту замовнику	30.05.2019	

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

6.1 Види випробувань

Для контролю правильності роботи програмного забезпечення буде проведено функціональне тестування. В ході тестування буде проведено випробування основних функціональних характеристик застосунку та цілої моделі загалом.

Тестування реєстрації та авторизації полягає у введенні вірних або невірних даних.

Тестування побудови маршруту полягає у введенні точок маршруту, та відображення вже побудованого маршруту на мапі.

Тестування контролю недотримання маршруту полягає у повідомленні керівника в разі відхилення перевізника від маршруту.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО
Керівник проекту
_____ *О.В.Ковтунець*
(підпис) (ініціали, прізвище)
“13” травня 2019 р.

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“14” травня 2019 р.

Комплекс задач підтримки логістичної діяльності
малих підприємств

ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

Шифр *ДП ІС-5206.1181-с.ПМВ*

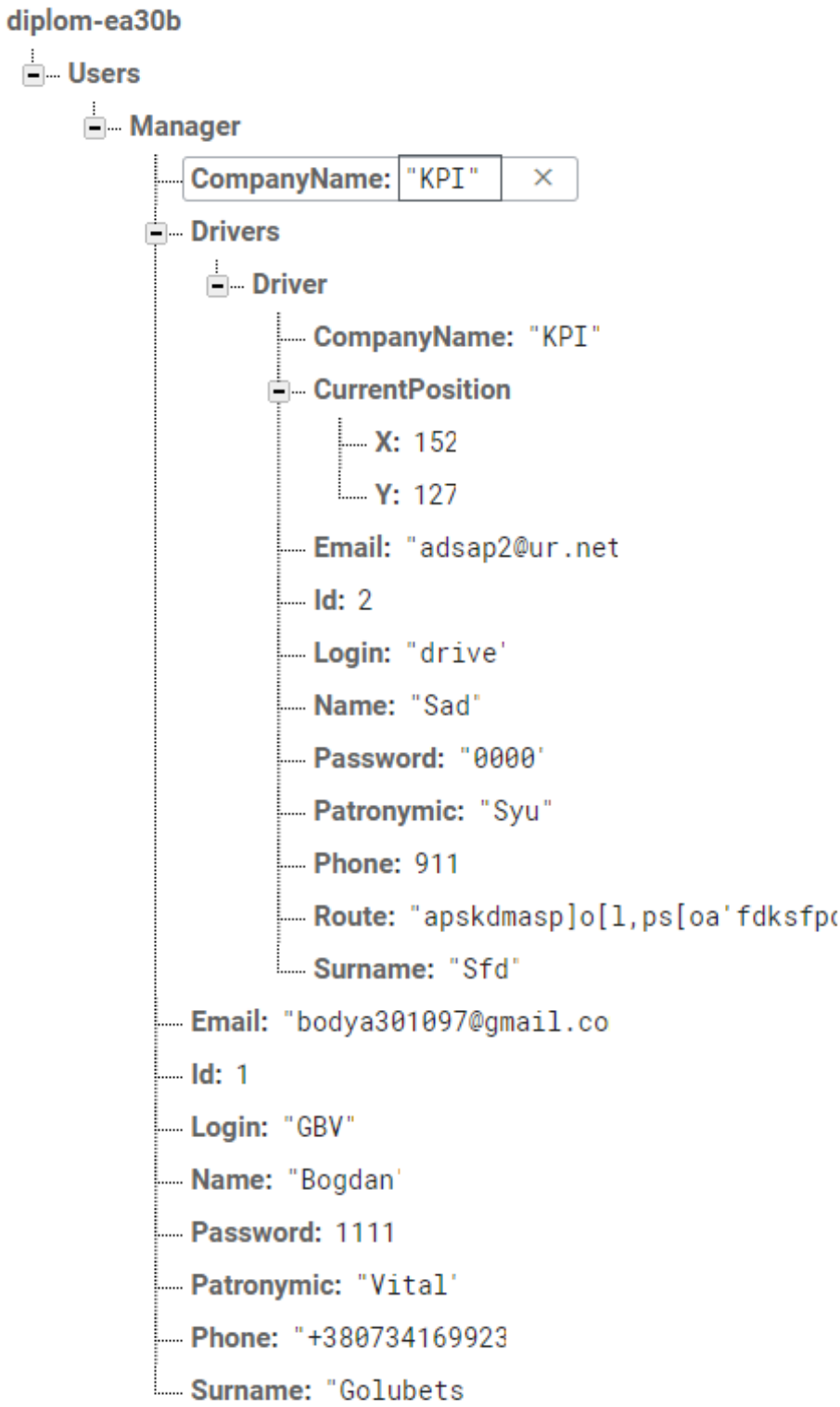
на 13 сторінках

Київ – 2019 року

Графічний матеріал
до дипломного проекту

на тему: Комплекс задач підтримки логістичної діяльності малих підприємств

Київ – 2019 року



					ДП ІС-5206.1181-с.СБД		
Зм.	Арк.	№ документа	Підпис	Дата	Схема бази даних	Літера	Маса / Масштаб
Розробив		Голубець Б.В.					
Перевірив		Ковтунець О.В.				Аркуш 1	Аркушів 1
Т. кон.					Комплекс задач підтримки логістичної діяльності малих підприємств	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52	
Н. кон.		Халус О.А.					
Затвердив		Ковтунець О.В.					

					ДП ІС-5206.1181-с.ССД		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна діяльності	Літера	Маса / Масштаб
Розробив		Голубець Б.В.					
Перевірив		Ковтунець О.В.				Аркуш 1	Аркушів 1
Т. кон.					Комплекс задач підтримки логістичної діяльності малих підприємств	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52	
Н. кон.		Халус О.А.					
Затвердив		Ковтунець О.В.					

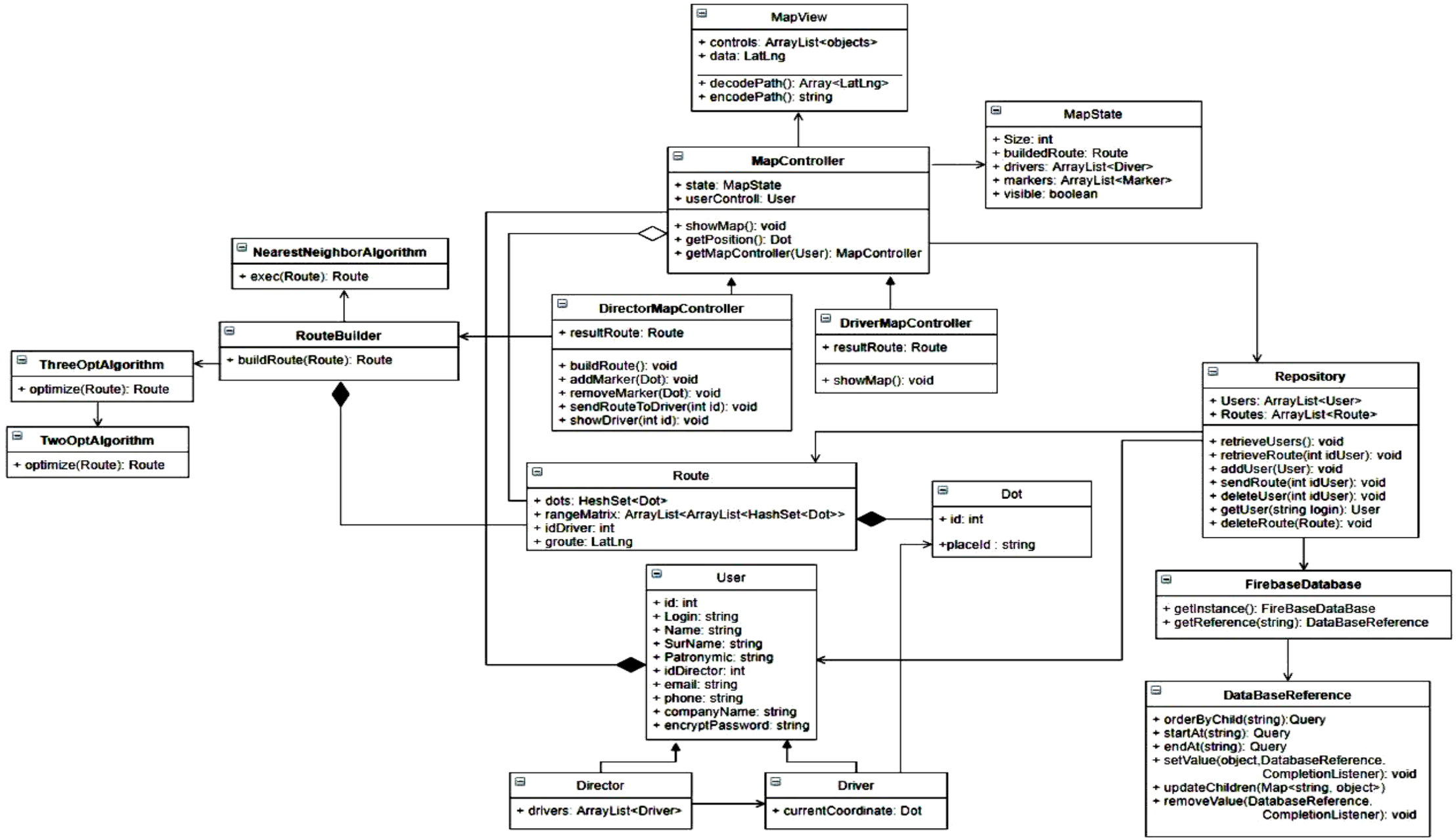
ДП ІС-5206.1181-с.ССК

					ДП ІС-5206.1181-с.ССК		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна компонентів програмного забезпечення	Літера	Маса / Масштаб
Розробив		Голубець Б.В.					
Перевірив		Ковтунець О.В.				Аркуш 1	Аркушів 1
Т. кон.					Комплекс задач підтримки логістичної діяльності малих підприємств	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52	
Н. кон.		Халус О.А.					
Затвердив		Ковтунець О.В.					

ДП ІС-5206.1181-с.ССВ

					ДП ІС-5206.1181-с.ССВ		
Зм.	Арк.	№ документа	Підпис	Дата	*Схема структурна варіантів використань*	Літера	Маса / Масштаб
Розробив		Голубець Б.В.					
Перевірив		Ковтунець О.В.				Аркуш 1	Аркушів 1
Т. кон.					*Комплекс задач підтримки логістичної діяльності малих підприємств*	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52*	
Н. кон.		Халус О.А.					
Затвердив		Ковтунець О.В.					

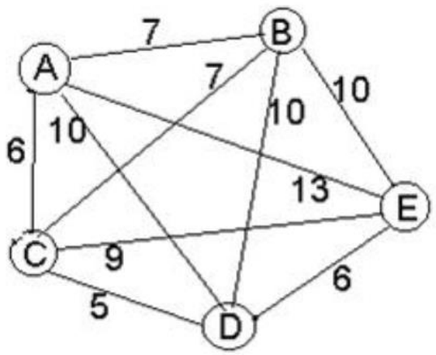
ДП ІС-5206.1181-с.ССК



					ДП ІС-5206.1181-с.ССК
Зм.	Арк.	№ документа	Підпис	Дата	*Схема структурна класів програмного забезпечення*
Розробив		Голубець Б.В.			Літера: — Маса: — Масштаб: —
Перевірив		Ковтунець В.О.			Аркуш 1 / Аркушів 1
Т. кон.					
Н. кон.		Халус О.А.			*Комплекс задач підтримки логістичної діяльності малих підприємств*
Затвердив		Ковтунець В.О.			*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52*

Рішення з математичного забезпечення

Дано:

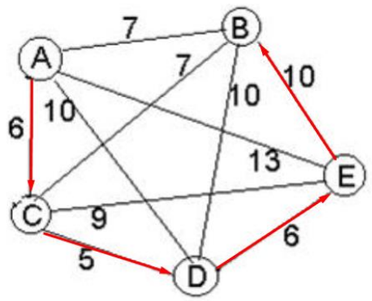


Необхідно знайти найкоротший шлях обходу усіх вершин при старті з вершини А, не відвідуючи раніше відвідані вузли

Алгоритм найближчого сусіда:

КРОК 1. Знайти найближчу невідвідану точку і перейти до неї;
КРОК 2. Якщо лишаються невідвідані точки, то перейти до кроку 1;
КРОК 3. Закінчити.

Результатом виконання алгоритму найближчого сусіда є початковий маршрут: A-C-D-E-B, його вага складає 27.



2 – Opt Алгоритм

Далі необхідно оптимізувати шлях

КРОК 1. Обрати 2 довільні ребра зі шляху, та видалити їх;
КРОК 2. Якщо вага маршруту зменшилась – перебудувати початковий шлях на знайдений;
КРОК 3. Якщо ще є пари ребер,зі шляху які не видаляли - перейти до кроку 1;
КРОК 4. Закінчити.

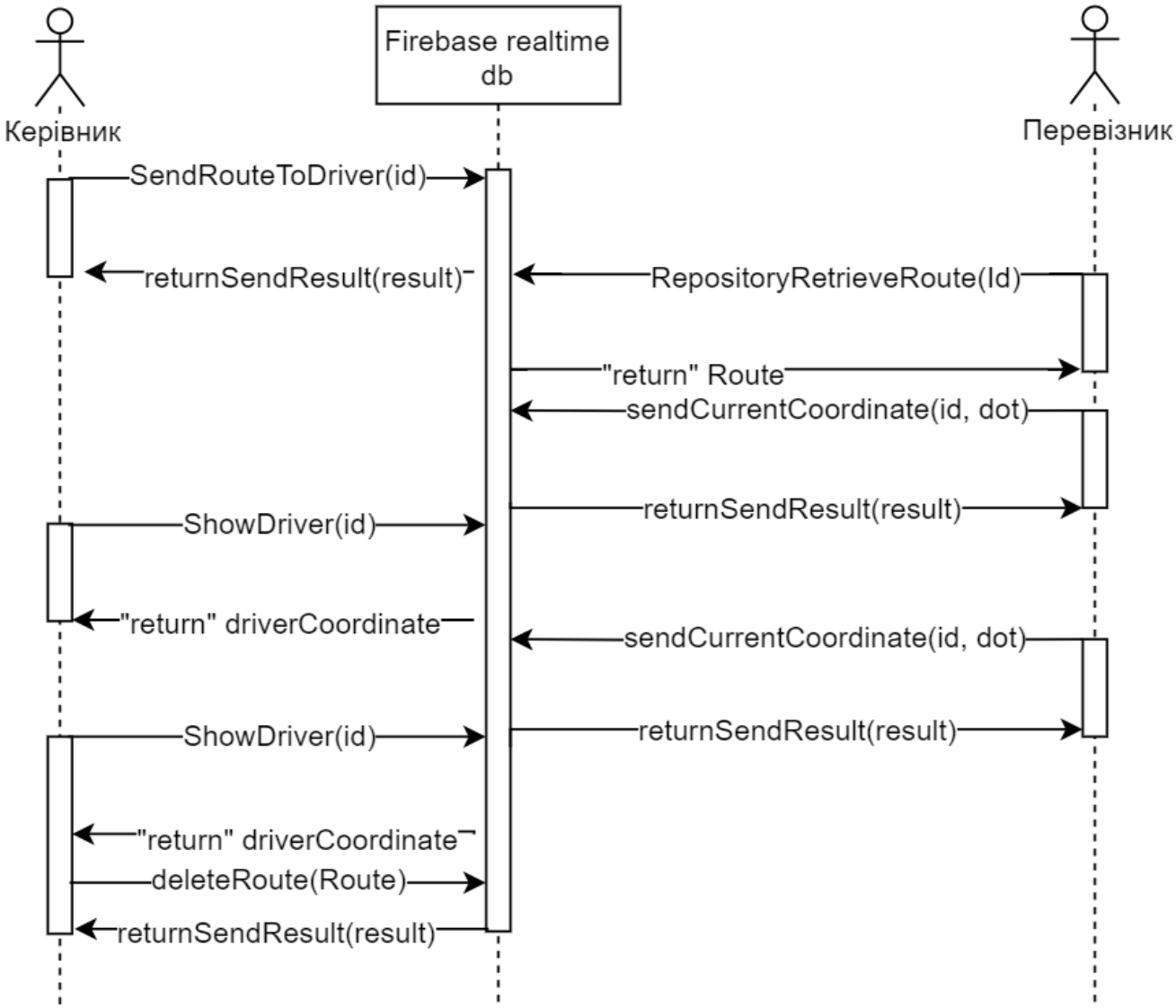
На завершення виконання 2-Opt Алгоритму не було знайдено шлях з меншою вагою, отже оптимальним залишився маршрут A-C-D-E-B, з вагою маршрута 27

Демонстраційний плакат до дипломного проекту

„Комплекс задач підтримки логістичної діяльності малих підприємств ”

Виконав студент гр. ІС-52 Голубець Б.В.

Керівник ДП Ковтунець О.В.



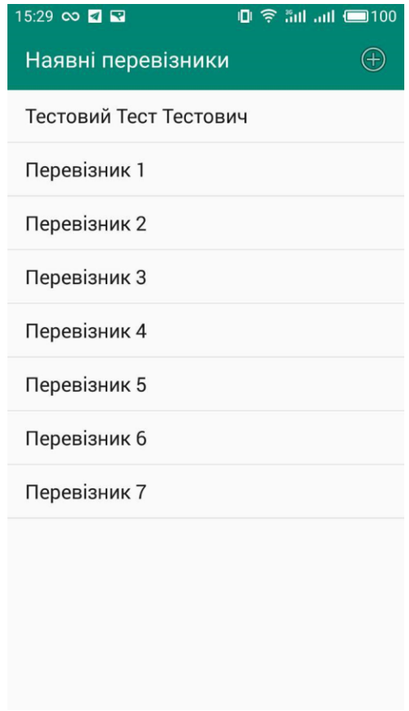
					ДП ІС-5206.1181-с.ССП			
						Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна послідовності			
Розробив		Голубець Б.В.				Аркуш 1		Аркушів 1
Перевірив		Ковтунець О.В.						
Т. кон.					Комплекс задач підтримки логістичної діяльності малих підприємств	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52		
Н. кон.		Халус О.А.						
Затвердив		Ковтунець О.В.						

СторІнка авторизації користувача

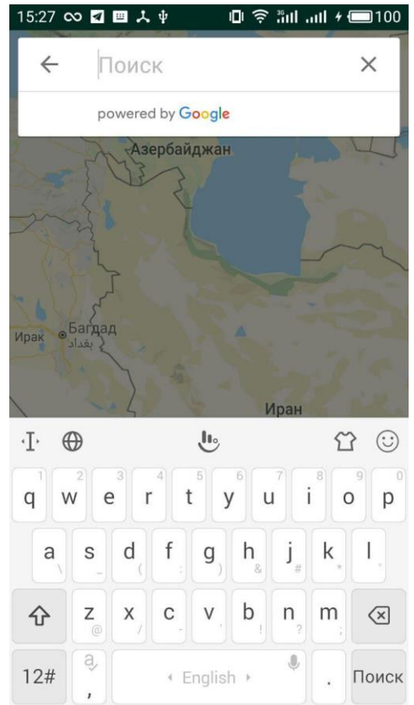
СторІнка реєсттрації користувача

Сторінка головного меню керівника

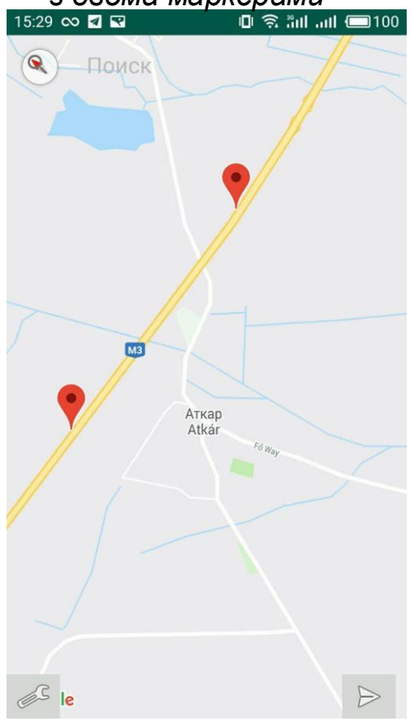
СторІнка редагування перевізників



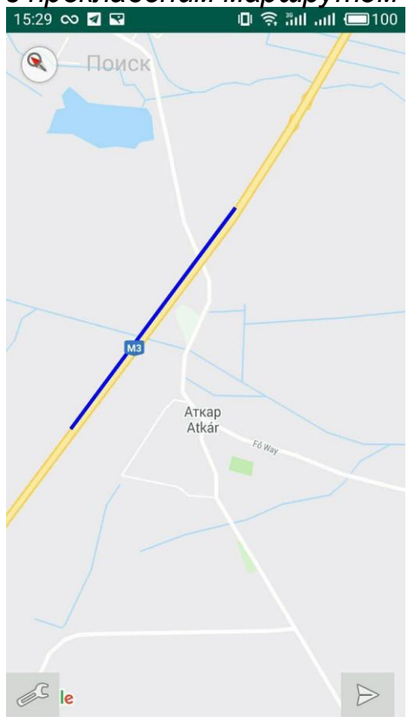
СторІнка мапи для побудови маршрута з пошуковим рядком



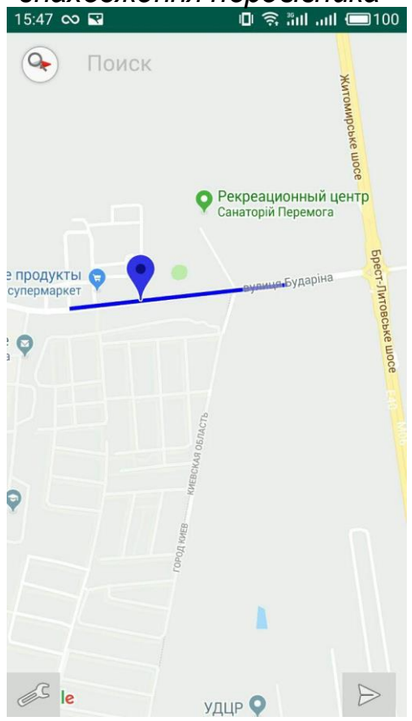
Сторінка мапи для побудови маршрута з двома маркерами



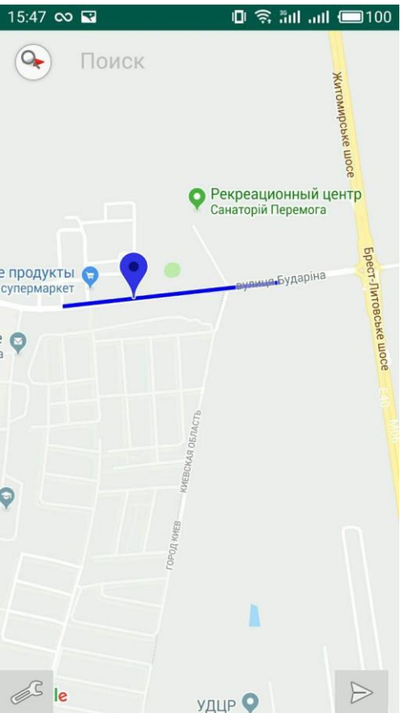
СторІнка мапи для побудови маршрута з прокладеним маршрутом



Сторінка мапи для контролю дотримання маршрута з місцем знаходження перевізника



					ДП ІС-5206.1181-с.КЕ	
Зм.	Арк.	№ документа	Підпис	Дата	*Креслення вигляду екранних форм*	Літера / Маса / Масштаб
Розробив		Голубець Б.В.				
Перевірив		Ковтунець О.В.				Аркуш 1 / Аркушів 1
Т. кон.						
Н. кон.		Халус О.А.			*Комплекс задач підтримки логістичної діяльності малих підприємств*	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52*
Затвердив		Ковтунець О.В.				


15:47
100
Поиск
Житомирське шосе
Рекреационный центр
Санаторій Перемога
е продукты
супермаркет
вулиця Будіна
Брест-Литовське шосе
КИЕВСКАЯ ОБЛАСТЬ
ГОРОД КИЕВ
УДЦР